ULTRIX

Guide to the BIND/Hesiod Service

Order Number: AA-LY21B-TE
June 1990

Product Version: ULTRIX, Version 4.0 or higher

The BIND/Hesiod service is a lookup service for information on host names, Internet Protocol addresses, and user and network services. This guide describes the BIND/Hesiod service.

digital equipment corporation
maynard, massachusetts

Contents

About This Guide

Audience vii
Organization vii
Related Documents viii
Conventions viii
New and Changed Information viii

1 Introduction to the BIND/Hesiod Service

1.1 BIND/Hesiod Servers 1–3
1.1.1 Root Servers 1–4
1.1.2 Master Servers 1–4
1.1.3 Caching Servers 1–5
1.1.4 Forwarding Servers 1–5
1.1.5 Slave Servers 1–6
1.2 BIND/Hesiod Clients 1–7
1.3 Resolving Queries 1–7
1.3.1 BIND Queries 1–7
1.3.2 Hesiod Queries 1–8

2 Format of BIND/Hesiod File Entries

2.1 Format of BIND/Hesiod File Entries 2–1
2.2 Description of Data File Entries 2–3
2.2.1 The include Entry 2–3
2.2.2 The origin Entry 2–3
2.2.3 The Address Entry 2–3
2.2.4 The Canonical Name Entry 2–4
2.2.5 The Host Information Entry 2–4
2.2.6 The Mailbox Entry 2–5

2.2.7 The Mail Group Data File Entry 2–5
2.2.8 The Mailbox Information Entry 2–6
2.2.9 The Mail Rename Entry 2–6
2.2.10 The Mail Exchanger Entry 2–7
2.2.11 The Name Server Entry 2–7
2.2.12 The Domain Name Pointer Entry 2–8
2.2.13 The Start of Authority Entry 2–8
2.2.14 The Hesiod Text Entry 2–9
2.2.15 The Well Known Services Entry 2–10

3 Configuring the BIND/Hesiod Service

3.1 The svc.conf File 3–1
3.1.1 Editing the svc.conf File with svcsetup 3–2
3.1.2 Editing the svc.conf File Manually 3–4
3.2 Using the bindsetup Command 3–4
3.2.1 Configuring a BIND/Hesiod Client 3–4
3.2.2 Configuring a BIND/Hesiod Primary Server 3–6
3.2.3 Configuring a BIND/Hesiod Secondary or Slave Server 3–8
3.2.4 Configuring a BIND Caching Server 3–9
3.3 Removing the BIND/Hesiod Service 3–11

4 Managing and Using the BIND/Hesiod Service

4.1 Maintaining the Domain 4–1
4.1.1 Domain Administrator Role 4–1
4.1.2 The Technical and Zone Contact 4–2
4.2 Naming Domains and Hosts 4–2
4.3 Registering With Public Networks 4–3
4.3.1 Contacting the DARPA Internet Network 4–3
4.3.2 Contacting the CSNET 4–3
4.3.3 Contacting the BITNET 4–3
4.4 Updating BIND/Hesiod Data Files on the Primary Server 4–4
4.5 Obtaining Host Name and IP Address Information 4–5
4.5.1 The nslookup Command 4–5
4.5.2 The nsquery Command 4–5
4.5.3 The NIC whois Service 4–6
4.6 Obtaining Further Information about the BIND Service 4–7
4.7 How to Make and Use a Hesiod Database 4–7

5 Troubleshooting the BIND/Hesiod Service

5.1 Reviewing the Domain Data Files 5–1

5.2 Reviewing the /etc/rc.local File 5–2

5.3 Reviewing the Resolver File 5–2

5.4 Reviewing the Hesiod File 5–2

5.5 Reviewing the Debug Files 5–2

5.5.1 The syslog File 5–3
5.5.2 The named_dump.db File 5–3
5.5.3 The named.run File 5–4
5.5.4 The named.stats File 5–4

5.6 Obtaining the named Process ID 5–5

5.7 Sending Signals to the named Daemon 5–5

A Configuring the BIND/Hesiod Service Manually

A.1 Configuring a BIND/Hesiod Client A–1

A.1.1 Create the resolv.conf File A–1
A.1.2 Create the hesiod.conf File A–2
A.1.3 Edit the hosts File A–2

A.2 Configuring a BIND/Hesiod Server A–2

A.2.1 Edit the Boot File A–2
A.2.2 Edit the Domain Data Files A–3
A.2.3 Edit the rc.local file A–3
A.2.4 Start the named daemon A–5

B Sample Files

B.1 Configuration Files B–2

B.1.1 Sample hesiod.conf file B–2
B.1.2 Sample resolv.conf file B–2
B.1.3 Sample svc.conf file B–2

B.2 Sample named.boot File B–3

B.2.1 Sample Primary Server Boot File B–3
B.2.2 Sample Secondary Server Boot File B–4
B.2.3 Sample Slave Server Boot File B–4
B.2.4 Sample Caching Server Boot File B–5

B.3 Sample named.ca File B–5

B.4 Sample named.local File B–5
B.5 Sample aliases.db File B–6
B.6 Sample auth.db File B–6
B.7 Sample group.db File B–6
B.8 Sample hosts.db File B–7
B.9 Sample hosts.rev File B–7
B.10 Sample networks.db File B–8
B.11 Sample passwd.db File B–8
B.12 Sample protocols.db File B–8
B.13 Sample rpc.db File B–9
B.14 Sample services.db File B–9
B.15 The named_dump.db File B–10
B.16 The named.run File B–10

C The nslookup Command

C.1 Getting nslookup Help C–1
C.2 Seeing Which nslookup Options Are Set C–2
C.3 Listing Hosts in a Domain C–2
C.4 Finding Mail Exchangers C–3
C.5 Finding the Start of Authority C–3
C.6 Looking at Hesiod Information in a Domain C–4
C.7 Finding Servers for a Domain C–4
C.8 Obtaining a Debug Trace C–5

D BIND Questionnaire

E References

Figures

1-1: Hierarchy of BIND Zones and Domains on the Internet 1–3
1-2: Relationship of Master/Forwarder and Slave Servers 1–6

About This Guide

The BIND/Hesiod service is a lookup service for information on host names, Internet Protocol addresses, and user and network services. This guide provides introductory information about the the Berkeley Internet Name Domain (BIND) service and the Hesiod name server, and explains how to install, configure, use, and troubleshoot systems running the BIND/Hesiod name service. In addition, it suggests guidelines for site-specific management procedures for running the BIND/Hesiod name service.

Audience

This guide is for the person who maintains networks and system utilities such as `mail` on the ULTRIX operating system. This person is usually the system manager, but could be a network manager or the system manager who is also a user of a VAX or RISC processor running the ULTRIX operating system. This guide assumes that you are familiar with the ULTRIX system commands, the system configuration, the naming conventions, and an editor such as `vi` or `ed`. It also assumes that you know the names and addresses of the other systems on the local network.

Organization

This guide consists of five chapters, five appendixes, and an index:

Chapter 1	Introduces the BIND service and the Hesiod name server, which is layered on top of BIND and provides background information needed to set up and run the BIND/Hesiod service on your system.
Chapter 2	Describes the format of files used by the BIND/Hesiod service.
Chapter 3	Describes how to use the `bindsetup` command to configure the BIND/Hesiod service on server and client systems.
Chapter 4	Describes how to manage the BIND/Hesiod service including: defining the BIND administrative roles, how to register a new top-level domain, and how to use some of the BIND features.
Chapter 5	Describes how to debug the BIND/Hesiod service and review the resulting debug files. It also offers general suggestions on how to troubleshoot the BIND/Hesiod service.
Appendix A	Describes how to configure the BIND/Hesiod service manually on server and client systems.
Appendix B	Shows sample configuration and data files that the BIND/Hesiod service uses to resolve queries.
Appendix C	Shows sample interactive sessions with the `nslookup` command.
Appendix D	This appendix shows the BIND questionnaire that you must complete and send to the Network Information Center (NIC) domain registrar to register your BIND domain.

Appendix E Lists other papers, articles, and Request for Comment (RFC) papers that contain useful information about BIND.

Related Documents

You should have the related hardware documentation for your system available as well as the other system and network documents in the ULTRIX documentation set.

Conventions

The following conventions are used in this guide:

%	The default user prompt is your system name followed by a right angle bracket. In this manual, a percent sign (%) is used to represent this prompt.
#	A number sign is the default superuser prompt.
user input	This bold typeface is used in interactive examples to indicate typed user input.
rlogin	In syntax descriptions and function definitions, this typeface is used to indicate terms that you must type exactly as shown.
[ ]	In syntax descriptions and function definitions, brackets indicate items that are optional.
{ | }	In syntax descriptions and function definitions, braces enclose lists from which one item must be chosen. Vertical bars are used to separate items.
. . .	In syntax descriptions and function definitions, a horizontal ellipsis indicates that the preceding item can be repeated one or more times.
. . .	A vertical ellipsis indicates that a portion of an example that would normally be present is not shown.

New and Changed Information

This manual is a revision. New and changed information includes the following:

- The title of the guide is now *Guide to the BIND/Hesiod Service.*
- Information on the Hesiod name server is incorporated throughout the manual.
- Material on setting up the BIND/Hesiod service manually is in Appendix A.
- Pathnames and file names have been updated.

Introduction to the BIND/Hesiod Service 1

This chapter provides an overview of the Berkeley Internet Name Domain (BIND) service and of the Hesiod name server. The BIND service is a host name and address lookup service for the Internet network. It allows client systems to obtain host names and addresses from BIND servers.

You can use the BIND service to replace or supplement the host table mapping provided by the local `/etc/hosts` file or the Yellow Pages (YP) service. The BIND service has two parts, the software interface and the server.

The software interface is called the resolver, which consists of a group of routines that reside in the C library `/usr/lib/libc.a`. The resolver exchanges query packets with a BIND server.

All BIND servers have a name server daemon, `named`, running in the background, which services queries on a given network port. The standard port for UDP and TCP is specified in the `/etc/services` file.

You can run BIND in conjunction with the Hesiod name server. Hesiod is layered on top of BIND and provides a name service for network objects that are used by workstations and timesharing systems. Hesiod allows an application to retrieve user-defined associations between a name, a particular type of service, and information about that service. For example, Hesiod can distribute information about network services, and host and user information.

By using the Hesiod name server you can replace or supplement the following databases:

- aliases
- auth
- group
- networks
- passwd
- protocols
- rpc
- services

To understand how the BIND/Hesiod service works, you must be familiar with Internet Protocol (IP) addressing. For a discussion of IP addresses, see the *Guide to Networking*.

The BIND service breaks the Internet into a hierarchy of domains, similar to a tree structure. Each domain is given a label. The name of the domain is the concatenation of all the labels of the domains, from the root to the current domain, listed from right to left and separated by dots. A label must be unique within its domain.

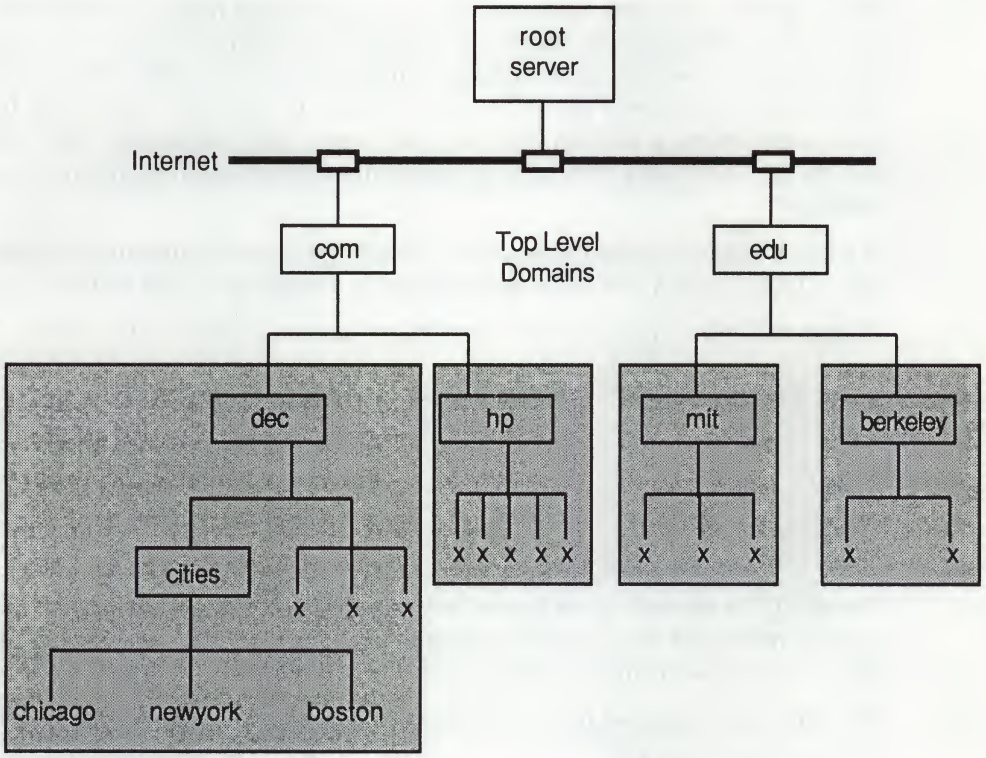
The entire BIND Internet hierarchy is partitioned into several zones, each starting at a domain and extending down to the leaf domains (individual host names), or to domains where other zones start. A zone is a subdivision of a domain and is a discrete, non-overlapping entity. Each zone is an area of authority for which a master server is responsible. See Section 1.1.2 for a discussion of master servers. Zones usually represent an administrative boundary.

The BIND hierarchy in the United States contains seven top-level domains:

arpa	For the Arpanet (gradually being phased out)
com	For commercial institutions
edu	For educational institutions
gov	For the government
mil	For military organizations
net	For network-type organizations, such as network service centers, consortia, and information centers
org	For miscellaneous organizations, such as professional societies and similar non-profit organizations

In addition to these, there are several top-level domains for individual countries. Contact the Network Information Center (NIC) for more information about them. Figure 1-1 shows the hierarchy of the Internet, two top-level domains, and some of the major zones. For example, in Figure 1-1, everything below `com` is in the `com` top-level domain; the zones are shaded. The leaf domains (host names) have the names of cities or are depicted by an `x`.

Figure 1-1: Hierarchy of BIND Zones and Domains on the Internet



ZK-0013U-R

Assuming a host name in the zone `cities.dec.com` is `chicago`, the following is the fully qualified domain name for that host:

```
chicago.cities.dec.com.
```

In this example, `com` is the top level domain, `cities.dec.com` is a subdomain of `com`, and `chicago` is a host name. If a master server has the authority for the `dec.com` domain only, then `dec.com` is a zone.

The dot (.) at the end of the domain name indicates that the domain name is fully qualified, and is absolute. No further BIND extensions should be appended to the name.

1.1 BIND/Hesiod Servers

The BIND/Hesiod service is based on a server-client model. Servers maintain databases of host names and addresses, and network objects. When client systems require the information, they query the servers.

A BIND/Hesiod server is a system running the `named` daemon. BIND/Hesiod has the following types of servers: root, master, caching, forwarding, and slave. The following sections describe each one in detail.

1.1.1 Root Servers

Root servers know about all the top-level domains on the Internet network. From these top-level domains, information is gathered about hosts on subdomains. The root servers, for example, do not necessarily know about the `cities.dec.com` subdomain. However, by performing a name server query with the `nslookup` command, a root server can tell you to check with `decwrl.dec.com` for information about a host on the `cities.dec.com` subdomain. See Appendix C and the `nslookup`(1) reference page for more information on the `nslookup` command.

If a client requests information about a host in a domain other than its own, any server (other than a slave) can pass along the request to a root server.

At this time there are seven root servers in the continental United States. They are:

```
ns.nasa.gov.
nic.ddn.mil.
a.isi.edu.
gunter-adam.af.mil.
aos.brl.mil.
terp.umd.edu.
c.nyser.net.
```

The dot (.) at the end of each root server name indicates that this is the absolute domain name and that no BIND name extensions are to be appended. Without the dot (.), the server name is relative to the current domain.

The NIC determines who are root servers. The toll-free number for the NIC is:

```
(800)235 - 3155
```

The electronic mailing address is:

```
hostmaster@nic.ddn.mil
```

1.1.2 Master Servers

A master server is the authority for the current domain space and maintains the BIND/Hesiod databases for its zone. A server can be a master server for multiple domains, being the primary server for some domains and a secondary server for others.

The primary server loads its database from a file on disk. This server can also delegate to other servers in its zone the authority to answer queries for its domain space.

A secondary server receives its authority and its database from the primary server. When a secondary server first boots, it loads the data for the zone from a backup file, if possible (assuming you configured your BIND/Hesiod service this way). It then consults a primary server to check that the database is still up to date.

After the secondary server is running, it periodically checks with the primary server to see if any database information has changed. If the database files have been modified, it updates its information. By default, the secondary servers poll the master server every five minutes to be sure that their database information is current. For information on how to define the frequency of the update checks, see Section 2.2.13.

Note

Because the secondary servers poll the master server at predetermined intervals, there is a time lag between when changes are made in the master server's databases and when they are pulled over to the secondary servers.

Each BIND/Hesiod domain should have at least two master servers, one primary and one or more secondary. The secondary servers act as backup servers in the event that the primary server fails, is overloaded, or is down.

1.1.3 Caching Servers

All servers cache the information they receive for use until the data expires. However, caching servers have no authority for any zone, and thus have no databases to maintain. These servers service BIND queries by asking other servers who have authority, such a master server, for the information. Caching servers store the information until it expires. The expiration is based on a time-to-live (`ttl`) field, which is attached to the data when the caching server receives it.

Note

If you set up a caching server, you might not receive Hesiod information for your local domain. Caching servers send queries to the root servers for resolution and the root servers might not know how to answer.

1.1.4 Forwarding Servers

Forwarding servers, called forwarders, process recursive requests that slave servers cannot resolve locally. A forwarder can be any BIND/Hesiod server that has Internet access. Thus, a forwarder can be a primary or a secondary server or a caching server. The configuration files on the slave servers define which systems the slaves access as forwarders.

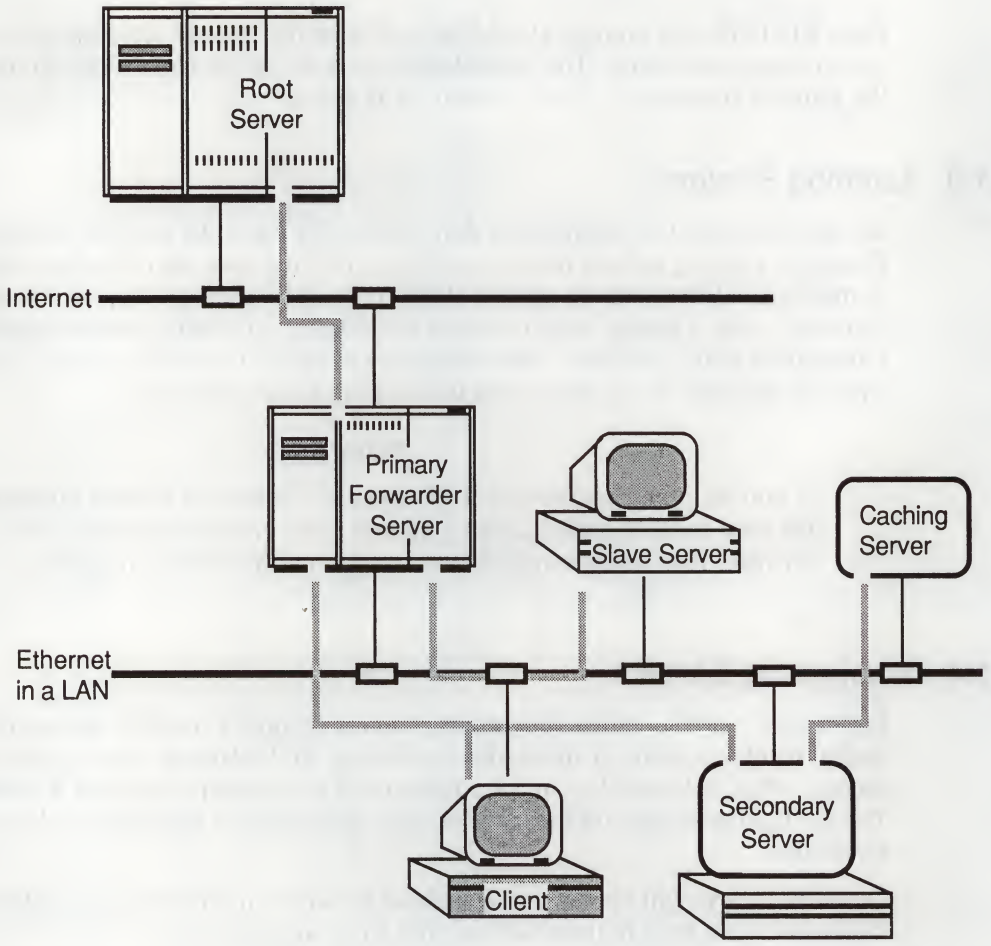
Forwarders have full access to the Internet network and therefore are able to obtain information not held in their local caches from root servers.

Because forwarders receive many requests from slave servers, they tend to have a larger local cache than do slave servers. All the hosts on the domain benefit from this larger cache, which reduces the total number of queries from that site by forwarding them to the root servers on the outside Internet network.

A slave server and forwarder configuration is typically used when you do not want all the servers to interact with the rest of the servers on the Internet network. For example, assume a site consists of several workstations that do not have access to the Internet network, and a VAX 8800 processor acting as a BIND/Hesiod forwarder. To give the workstations the appearance of access to the Internet network, you could set them up as BIND/Hesiod slave servers to the VAX 8800 system. In this case, the BIND/Hesiod forwarder forwards the workstations' queries and interacts with other BIND/Hesiod servers on the Internet network. When the forwarder resolves the queries, it sends the answer to the slave server.

Figure 1-2 shows the relationship between root servers, master servers, slave servers, forwarding servers and clients.

Figure 1-2: Relationship of Master/Forwarder and Slave Servers



You can run the BIND/Hesiod service on a local network only, without having a forwarder on your network. However, if you do not have a forwarder on your network, there is no need to have slave servers. Without forwarders, your system does not have access to the root servers on the Internet.

1.1.5 Slave Servers

Slave servers do not have access to the Internet, and therefore cannot interact directly with root servers to get information that is not in their local cache. Instead, slave servers use forwarders, which can be either primary or secondary servers, to resolve their queries. A slave server forwards queries to the list of forwarders specified in its boot file, until the list is exhausted or the query is satisfied.

As the slave servers request new information from forwarders, they accumulate it in their cache. Slave servers do not receive complete zones from primary servers, like a secondary server does, but accumulate data per request.

1.2 BIND/Hesiod Clients

A BIND/Hesiod client is any system that uses the BIND/Hesiod service to resolve host names and addresses, and user and network service information. BIND/Hesiod clients make queries, but they never resolve them locally. Instead, BIND/Hesiod servers resolve the clients' requests.

BIND/Hesiod clients do not run the `named` daemon. Instead, BIND/Hesiod clients have the resolver file, `/etc/resolv.conf`, which tells the resolver the IP address of the BIND/Hesiod servers which can service the client's BIND/Hesiod requests. Here is an example of a `/etc/resolv.conf` file:

```
domain      dec.com
nameserver 128.11.22.33
nameserver 128.11.22.44
```

1.3 Resolving Queries

This section describes the steps that the BIND/Hesiod service takes to resolve queries about host names or network objects.

Hesiod queries are ultimately passed to BIND in a BIND-compatible format.

1.3.1 BIND Queries

The following steps describe the usual procedure a slave server and forwarder take to resolve a BIND query. The process is similar for other servers:

1. A slave server receives a query for a host name resolution.
2. The slave server uses the `gethostbyname` library routine.
3. The `gethostbyname` library routine calls the naming services listed for `hosts` in the `/etc/svc.conf` file. When the library routine calls BIND, the slave server asks the forwarders listed in its BIND boot file (the default is `named.boot`) one at a time, until the query is resolved or the list is exhausted.
4. If the forwarder does not have the information in its local cache, it asks the root servers listed in its BIND data file, one at a time, until the query is resolved or the list is exhausted.
5. The root server provides the forwarder with the information needed to contact servers of the domain space containing the host in question.
6. The forwarder sends a request to a server for that domain. It gets the server's address information from a root server.
7. The server provides the forwarder with the information to contact servers of the next lower domain.
8. Steps 5 and 6 repeat until the forwarder actually gets the host information, or until the information obtained from the root server is exhausted.
9. The forwarder returns the results to the slave server, even if the results show the resolution was not successful.

1.3.2 Hesiod Queries

A Hesiod query consists of two parts:

- *HesiodName*
- *HesiodNameType*

The *HesiodName* is the name of an object, such as a group name, service name, or user name. It consists of two strings, a left-hand string (LHS) and a right-hand string (@RHS).

Hesiod object names can contain dots (.), which BIND does not permit, allowing for greater flexibility in naming objects.

The *HesiodNameType* is a well-known string that is provided by an application that uses the Hesiod library. It identifies the query to BIND, and the proper expansion rules to use with the LHS and RHS of the name. The expansion rules for the LHS and RHS are defined in the `/etc/hesiod.conf` file. Although definitions for both the LHS and RHS can be present, the RHS is required. This file is mandatory for all systems that are configured to use the BIND/Hesiod service.

When an application passes a query to Hesiod, the Hesiod library takes the arguments and translates them, using the `hes_resolve` and `hes_to_bind` subroutines, into valid BIND strings. The strings are then passed to the BIND resolver library and, once resolved, are passed back to the Hesiod name server, which passes the information to the calling application.

Format of BIND/Hesiod File Entries 2

The boot file, by default called `/var/dss/namedb/named.boot`, specifies the names of the BIND/Hesiod data files. These data files consist of entries, also known as Resource Records (RR), that follow the formats described in this chapter. The sample files in Appendix B contain some of these formats.

2.1 Format of BIND/Hesiod File Entries

Here is the general format of a BIND/Hesiod data file entry (RR):

name ttl addr-class entry-type entry-specific-data

The fields are as follows:

name

This is the name of the domain, for example `cities.dec.com`. The domain name must begin in the first column.

For some data file entries the name field is left blank. In that case, the domain name is assumed to be the same as the previous entry.

A free standing period (.) refers to the current domain.

A free standing at sign (@) denotes the current origin, thus allowing you to specify more than one domain.

Two free standing periods (..) represent the null domain name of the root.

ttl

This is the time-to-live field, and specifies how long, in seconds, the data will be stored in the database. If this field is left blank, the value defaults to the `ttl` specified in the SOA (start of authority) entry. The maximum time-to-live is 99999999 seconds, or 3 years.

addr-class

This field is the address class. There are three classes:

IN -- Internet addresses

HS -- Hesiod naming service data

ANY -- All other types of network address

The address class of all data file entries of a given entry-type in a particular zone must be the same.

Therefore, only the first entry in a zone need specify the `addr-class` field.

entry-type

This field states the resource record type, for example `SOA` or `A`.

entry-specific-data All fields after the `entry-type` field vary for each type of date file entry (resource record).

The case is preserved in name and data fields when loaded into the BIND/Hesiod server. All comparisons and lookups using the BIND/Hesiod service are performed in a case insensitive manner.

The following characters have special meanings in BIND/Hesiod data file entries:

\x A backslash (\) escapes the next non-digit (*x*) character so that the character's special meaning does not apply. For example, you could use . to place a period character in a label.

\nnn A backslash denotes the octet corresponding to the decimal number represented by *nnn*. The resulting octet is assumed to be text and is not checked for special meaning.

( ) Parentheses group data that cross a line. In effect, line terminations are not recognized within parentheses.

; A semicolon starts a comment, causing the rest of the line to be ignored.

* An asterisk signifies a wildcard.

Most BIND/Hesiod data file entries have the current domain appended to their names if they are not terminated by a period (.). This is useful for appending the current domain name to the data, such as system names, but could cause problems when you do not want this to happen. Consequently, if the name is not in the domain for which you are creating the data file, end the name with a period.

Data files (resource records) can have the following types of entries:

- $include
- $origin
- A – address
- CNAME – canonical name
- HINFO – host information
- MB – mail box
- MG – mail group
- MINFO – mailbox information
- MR – mail rename
- MX – mail exchanger
- NS – name server
- PTR – domain name pointer
- SOA – start of authority
- TXT – Hesiod data or text
- WKS – well know services

2.2 Description of Data File Entries

The following sections describe each data file entry and its format.

2.2.1 The include Entry

An `include` entry is similar to a header file in the C programming language. This feature is particularly useful for separating different types of data into multiple files. An `include` entry begins with `$include` in the first column, and is followed by the name of the file to be included. For example:

```
$include /var/dss/namedb/mailboxes
```

This entry requests the BIND/Hesiod to load the data file `$include /var/dss/namedb/mailboxes`

The `$include` entry loads data files into the local zone and acts as a data file organizer. For example, you can use `$include` entries to separate mail from host information.

2.2.2 The origin Entry

An `origin` entry changes the origin in a data file. This feature is particularly useful for putting more than one domain in a data file. An `origin` entry begins with `$origin` in the first column, followed by a domain origin. For example:

```
$origin state.dec.com.
```

This entry includes the domain `state.dec.com` in the data file. As a result, the BIND/Hesiod can provide information about the `state.dec.com` domain in addition to the local domain, provided your server is authoritative for the zone.

The `$origin` and `$include` entries can work together. They can also save typing and help keep the files organized. For example, assume that the following entries are in the `hosts.rev` file:

```
$origin 11.128.in-addr.arpa.
$include cities.dec.com.rev
```

The period after `arpa` signifies the complete domain name.

Assume that the `cities.dec.com.rev` file consists of entries similar to the following:

```
33.22 IN  PTR chicago.cities.dec.com.
```

In this situation, the complete reverse name for the host `chicago` is translated to be as follows:

```
33.22.11.128.in-addr.arpa.  IN PTR chicago.cities.dec.com.
```

2.2.3 The Address Entry

The address (A) data file entry lists the address for a specific system. Here is the format for an A entry:

name ttl addr-class entry-type address

The fields in the A entry have the values described in Section 2.1, with the exception of the `address` field. This field specifies the IP address for each system. There should only be one A entry for each address of a given system.

Here is an example of two A entries:

```
;name                  ttl     addr-class    entry-type    address
miami.cities.dec.com.          IN            A             128.11.22.44
                               IN            A             128.11.22.33
```

In this example note that the first entry has left the ttl field blank, thus using the default ttl specified in the SOA entry. The second entry has left the first and second fields blank, thus using the default name specified in the previous entry and the default ttl specified in the SOA entry. In this example, the host `miami.cities.dec.com` has two IP addresses.

2.2.4 The Canonical Name Entry

The canonical name (CNAME) entry specifies an alias for a canonical name. For example, if the canonical name, (also known as the full BIND name or the fully qualified name) is `miami.cities.dec.com`, a reasonable alias might be `miami` or `mi`.

An alias must be unique, and all other entries should be associated with the canonical name and not with the alias. Do not create an alias and then use it in other entries. Here is the format of a CNAME entry:

aliases ttl addr-class entry-type can-name

The fields in the CNAME entry have the values described in Section 2.1, with the following exceptions:

aliases This field specifies the nickname (alias) of the canonical name of the host.

can-name This is the canonical name of the host. If the canonical name is a part of the current domain, you only need to specify the host name, for example, `miami`. If the canonical name is for a host in another domain, you must specify the fully qualified BIND name, followed by a period (.). For example: `ohio.state.dec.com`.

The following example shows two CNAME entries. The first entry is for a CNAME in the current domain, `cities.dec.com` ; the second entry is for a CNAME in another domain:

```
;aliases    ttl addr-class  entry-type  can-name
to              IN          CNAME       toledo
oh              IN          CNAME       ohio.state.dec.com.
```

2.2.5 The Host Information Entry

The host information (HINFO) data file entry is for host specific information. This entry lists the hardware and operating system that are running at the specified host system. Only a single space separates the name of the hardware from the operating system information. Thus, if you need to use spaces as part of a host or operating system name, you must place the name in quotation marks. In addition, there can be no more than one HINFO entry for each host on the domain. Here is the HINFO entry format:

host ttl addr-class entry-type hardware opsys

The fields in the HINFO entry have the values described in Section 2.1, with the following exceptions:

host This field specifies the host name. If the host is in the current domain, you only need to specify the host, say `chicago`, for example. If the host is in a different domain, you must specify the full BIND name, for example, `utah.state.dec.com.`. Be sure to include the period (.) at the end of the host name. This indicates the fully qualified BIND name.

hardware This field specifies the type of CPU, for example, a VAX 8800 processor.

opsys This field specifies the type of operating system running on the specified host and should be `ULTRIX` for the ULTRIX operating system.

Here is an example of a HINFO entry:

```
;name                ttl     addr-class  entry-type  hardware     opsys
ohio.state.dec.com.          IN          HINFO       VAX-11/780     ULTRIX
```

In this example, note that the second field specifying the ttl is blank, thus using the default ttl specified in the SOA entry.

2.2.6 The Mailbox Entry

The mailbox (MB) entry lists the system where a user wants to receive mail. Here is the format of an MB entry:

login ttl addr-class entry-type system

The fields in the MB entry have the values described in Section 2.1, with the following exceptions:

login This field is the login name for a user. Login names must be unique for the domain.

system This field specifies the name system where the user wants to receive mail.

Here is an example of an MB entry:

```
;login      ttl     addr-class     entry-type     system
fred                IN             MB             potsdam.cities.dec.com.
```

In this example note that the second field is left blank, thus using the ttl specified in the SOA entry. Consequently, the user `fred` will have mail delivered to the host named `potsdam` in the domain `cities.dec.com`.

2.2.7 The Mail Group Data File Entry

The mail group entry specifies the members of a mail group. The MG entry is usually used with a MINFO entry. Here is the format of an MG entry:

group ttl addr-class entry-type member

The fields in the MG entry have the values described in Section 2.1, with the following exceptions:

group This field specifies the name of the mail group, for example, `users` or `marketing`.

member This field specifies the login name and the domain of the user to be included in the mail group.

Here is an example of a MINFO entry and several MG entries:

```
;group  ttl addr-class  entry-type  requests        member
fun         IN          MINFO       BIND-REQUEST    fred@miami.cities.dec.com.
            IN          MG                          john@miami.cities.dec.com.
                        MG                          amy@miami.cities.dec.com.
```

In this example, note that the second field for all three entries is left blank, thus using the `ttl` specified in the `SOA` entry. In addition, if mail is sent to the mail group `fun`, fred, john, and amy receive it.

2.2.8 The Mailbox Information Entry

The mailbox information (MINFO) entry creates a mail group for a mailing list. The MINFO entry is usually associated with a mail group (MG) entry, but can also be used with a mailbox (MB) entry. Here is the format of a MINFO entry:

mailbox ttl addr-class entry-type requests maintainer

The fields in the MINFO entry have the values described in Section 2.1, with the following exceptions:

mailbox This field specifies the name of the mailbox, and is usually `BIND`.

requests This field specifies the name where users should send mail relating to the BIND/Hesiod or mail. For example, a user might want to send a mail message requesting that an alias be set up.

maintainer This field contains the login name of the person who should receive mail error messages. This is particularly useful when an error in member's names should be reported to a person other than the sender.

Here is an example of a MINFO entry:

```
;mailbox  ttl addr-class  entry-type  requests      maintainer
BIND          IN          MINFO       BIND-REQUEST  fred@miami.cities.dec.com.
```

In this example, note that the second field is left blank, thus using the ttl specified in the SOA entry.

2.2.9 The Mail Rename Entry

The mail rename (MR) entry lists aliases for a specific user. Here is the format of an MR entry:

alias ttl addr-class entry-type login

The fields in the MR entry have the values described in Section 2.1, with the following exceptions:

alias This field lists the nicknames for the specified user. The alias must be unique to the domain.

login This field is the login name for the user whose alias is being established. There should also be a corresponding MB entry for the specified login name.

Login names must be unique for the domain.

Here is an example of an MR entry:

```
;alias  ttl addr-class  entry-type  login
lady        IN          MR          diana
princess    IN          MR          diana
```

This example shows how to set up the aliases `lady` and `princess` for a user whose login name is `diana`. Note that the second field is left blank, thus using the ttl specified in the SOA entry.

2.2.10 The Mail Exchanger Entry

The mail exchanger (MX) entry specifies a system in the local domain (called a gateway) that knows how to deliver mail to a system that may not be directly connected to the local network. Consequently, the MX entry is useful for systems outside your local network that want to send mail to a user on one of your network's hosts.

You can also use the MX entry to list some of the hosts in the `/etc/hosts` file so that they do not appear to other systems using the BIND service.

Here is the format of an MX entry:

system ttl addr-class entry-type pref-value gateway

The fields in the MX entry have the values described in Section 2.1, with the following exceptions:

system This field specifies the name of the system where mail is to be sent.

pref-value This field specifies the order a mailer should follow when there is more than one way to deliver mail to a given system.

gateway This field contains the name of the gateway system, that is, the system that can deliver mail to the destination system on another network.

Here is an example of two MX entries:

```
;system                ttl addr-class  entry-type  pref-value  gateway
tampa.cities.dec.com       IN          MX          0           seismo.cs.au.
*.folks.dec.com            IN          MX          0           relay.cs.net.
```

In this example, all mail destined for the domain `folks.dec.com`, regardless of the host name, is sent by route of the `relay.cs.net.` host. In addition, note that the second field in both entries is left blank, thus using the ttl specified in the SOA entry. The second entry uses an asterisk, which is a wildcard.

2.2.11 The Name Server Entry

The name server (NS) entry specifies that a system is a name server for the specified domain. Here is the format of the NS entry:

name ttl addr-class entry-type server

The fields in the NS entry have the values described in Section 2.1, with the exception of the `server` field. This field specifies the name of the primary master server for the domain specified in the first field.

Here is an example of an NS entry:

```
;name    ttl addr-class  entry-type  server
             IN          NS          utah.states.dec.com.
```

In this example note that the first and second fields are left blank, thus using the domain specified in a previous entry and the ttl specified in the SOA entry.

2.2.12 The Domain Name Pointer Entry

The domain name pointer (PTR) entry allows special names to point to some other location in the domain. PTR names should be unique to the zone. These entries are located on a primary server in the file `/var/dss/namedb/hosts.rev`. Here is the format of a PTR entry:

rev-addr ttl addr-class entry-type hostname

The fields in the PTR entry have the values described in Section 2.1, with the following exceptions:

rev-addr This field specifies the reverse IP address of the host. For example, if the host's address is `128.11.22.33`, the reverse address is `33.22.11.128`.

hostname This is the fully qualified BIND name of the host, for example, `miami.cities.dec.com`. Be sure to include the period (.) at the end of the host name if the host is not in the current domain.

Here is an example of two PTR entries:

```
;rev-addr              ttl addr-class  entry-type  hostname
33.22                        IN          PTR         chicago
66.55.44.121.in-addr.arpa.   IN          PTR         mail.peace.org.
```

In this example, the first entry is for a host whose IP host address is `22.33` in the current domain. The specified rev.addr (`33.22`) is meaningful assuming that a `$origin` entry exists. See Section 2.2.2 for a description of the `$origin` entry. If there is not a `$origin` entry, then the entire IP address, in reverse, must be specified.

The second entry is for a host in different domain (`mail.peace.org.`). As a rule, you should not do this because you are putting data in your server's cache for which your server is not authoritative. PTR entries and other resource records should be for hosts in your domain, only. The PTR entry sets up a reverse pointer for the host `mail.peace.org`.

2.2.13 The Start of Authority Entry

The start of authority (SOA) entry designates the beginning of a zone. There should be no more than one SOA entry per zone. Here is the format of an SOA entry:

name ttl addr-class entry-type origin person serial# refresh retry expire min

The fields in the SOA entry have the values described in Section 2.1, with the following exceptions:

origin This field is the name of the host on which the data file resides. This is usually a primary server.

person This field defines the login name and mailing address of the person responsible for the BIND/Hesiod running on the local domain.

serial# This field specifies the version number of the data file. The person editing the master files for the zone should increment the value in this field each time a change is made to the data within the file. The serial number being changed informs the secondary servers that there is new data to be obtained from the primary server. The maximum number is $2^{32} - 1$ after the decimal point.

The serial number field allows the BIND/Hesiod to determine which

of two copies of data files in a zone are more recent. Typically, the serial number field begins at one (1) and is incremented by one each time the original data file is modified. It is best to use whole integers.

refresh This field specifies how often, in seconds, a secondary BIND/Hesiod server is to check with the primary server to see if it needs to update its data files. If the data files are out of date (as indicated by a mismatch of serial number fields), they are updated with the contents of the master server's files.

The minimum refresh period is 30 seconds. If the `refresh` field is left blank, however, the data file is not dynamically updated.

retry This field specifies how often in seconds, a secondary BIND/Hesiod server will try to refresh its data files after a refresh failure has occurred while making the check. If a BIND/Hesiod server attempts to refresh the files and fails, it tries to refresh them again every so many seconds, as specified in the `retry` field.

expire This field specifies the upper limit, in seconds, that a secondary BIND/Hesiod server can use the data files in its cache before the data expires for lack of being updated, or before the BIND/Hesiod server checks to see if its cache needs to be updated.

min This field specifies the default time to live, in seconds, that a data entry can exist in the event that the `ttl` entry is left blank.

The following is an example of an SOA entry. The first line is a comment that shows the fields:

```
;name ttl addr-class  entry-type  origin              person
@     IN    SOA     utah.states.dec.com. hes.utah.states.dec.com. (
                      1       ; serial
                      3600    ; refresh every hr.
                      300     ; retry every 5 min.
                      3600000 ; expire in 1000 hrs.
                      86400 ) ; min. life is 24 hrs.
```

In this example note that the parentheses indicate to the BIND/Hesiod that this is a single entry. The ttl field is left blank, indicating that the default time to live specified in the min field (86400 seconds) is being used.

The semicolons allow comments for readability. In the example, the serial field is 1, the refresh field is 3600 seconds (once per hour), the retry field is 300 seconds (once per 5 minutes), the expire field is 3,600,000 seconds (1000 hours), and the minimum field is 86400 seconds (24 hours).

2.2.14 The Hesiod Text Entry

The Hesiod text (TXT) entry indicates an application-specific piece of text. The Hesiod name service uses these records to store information for Hesiod domains, such as the `aliases`, `auth`, `group`, `networks`, `passwd`, `protocols`, `rpc`, and `services` databases.

Here is the format of a TXT entry:

text-key ttl addr-class entry-type text

The fields in the TXT entry have the values described in Section 2.1, with the following exceptions:

text-key This field is the Hesiod text key name.

text This field defines the Hesiod data or text relative to an application.

In the following example, the `networks` database, which consists of the networks `east-coast` and `west-coast`, is being distributed by the Hesiod name server. The domain is named `networks.cities.dec.com`.

```
;text-key    ttl addr-class  entry-type   text
east-coast       HS          TXT          "east-coast:128.11.22"
west-coast       HS          TXT          "west-coast:128.11.33"
```

The network `east-coast` has the network number 128.11.22, and `west-coast` has the network number 128.11.33.

2.2.15 The Well Known Services Entry

The well known services (WKS) entry describes well known services supported by a particular protocol at a specified address. The services and port numbers are obtained from the list of services specified in the `/etc/services` file. Here is the format of a WKS entry:

name ttl addr-class entry-type address protocol services

The fields in the WKS entry have the values described in Section 2.1, with the following exceptions:

address This field specifies the IP address for each system. There should only be one WKS entry for each protocol at each address.

protocol This field specifies the protocol to be used, for example `TCP` or `UDP`.

Here is an example of two WKS entries:

```
;name    ttl addr-class  entry-type  address     protocol  services
             IN          WKS         128.32.0.4  UDP       who route
             IN          WKS         128.32.0.78 TCP      (echo talk
                                                          discard sunrpc sftp
                                                          uucp-path netstat host
                                                          systat daytime link
                                                          auth time ftp
                                                          nntp whois pop
                                                          finger smtp supdup
                                                          domain nameserver
                                                          chargen)
```

Note that the first and second fields of both entries in this example are blank, which indicates that they are using the domain name specified in a previous entry and the default ttl specified in the SOA entry. The services listed in the second entry are contained within parentheses and are thus interpreted as being one entry, even though they appear to be on several lines.

Configuring the BIND/Hesiod Service 3

The ULTRIX software provides the `bindsetup` command to automate setting up the BIND/Hesiod service on your system. This chapter explains how to use the `bindsetup` command to configure your system as a BIND/Hesiod client or server. The `bindsetup` command creates numerous data files. Samples of these data files are in Appendix B.

All of the system files, scripts, and databases related to the BIND/Hesiod service are located in the directory `/var/dss/namedb`, and its subdirectories.

The `bindsetup` command configures your system to run both BIND and Hesiod. You can, however, run BIND without Hesiod if all you want is host name and address information.

Although it is not recommended, you can configure the BIND/Hesiod service manually. For information, see Appendix A.

If you want the BIND/Hesiod service to resolve queries about other domains, you must register your domain. Chapter 4 describes how to register your domain with a public network.

The BIND/Hesiod service uses many different files to resolve queries. This chapter describes how to edit, create, and work with those files. The manner in which your system uses the BIND/Hesiod service to search for information is established in the `/etc/svc.conf` file. This chapter describes how to edit the `/etc/svc.conf` file, subsequently referred to as, `svc.conf`, to define the order of name service queries on your system.

In addition, this chapter describes how to remove the BIND/Hesiod service from your system. This chapter discusses the following:

- How to edit the `svc.conf` file
- How to use the `bindsetup` command
- How to remove the BIND/Hesiod service

3.1 The svc.conf File

The `svc.conf` file defines the order in which to query the name services running on your system. It is a mandatory system file that is created when you install the ULTRIX software. If you want to use the BIND/Hesiod service, you must edit the `svc.conf` file with the necessary database and service order information. The following is a typical entry in the `svc.conf` file:

```
passwd=local,bind
```

This entry tells the system to search first locally for password information. If it can not find the information locally, the system then queries a BIND/Hesiod server.

Note

It is recommended that you list `local` as the first service for all databases.

You can specify any of the following databases in the `svc.conf` file:

- aliases
- auth
- group
- hosts
- netgroup (served only by Yellow Pages)
- networks
- passwd
- protocols
- rpc
- services

See Appendix B for a sample `svc.conf` file; see `svc.conf`(5) in the ULTRIX Reference Pages for more information.

3.1.1 Editing the svc.conf File with svcsetup

The `svcsetup` command allows you to print and modify the database selections in the `/etc/svc.conf` file on the current system. You must modify this file when you are adding or removing a naming service, such as Yellow Pages or BIND/Hesiod.

Run the `secsetup` command if you want to change the security parameters. Changes take effect immediately. See `secsetup`(8) in the ULTRIX Reference Pages for more information about setting these parameters.

To run the `svcsetup` command, you must be logged on as superuser. Type the following:

```
# svcsetup
```

The `svcsetup` command then steps through the setup procedure in the following manner:

1. Displays a menu asking whether you want to modify the existing `svc.conf` file, print the default settings to the screen, or exit the `svcsetup` command.

   Select the `modify` option.

2. Lists the databases that you can modify. Each database is assigned a corresponding number by the system. Select the databases that you want to modify by listing the number of each database, followed by a space.

   For example, to modify the `aliases` and `group` databases, enter their numbers as your repsonse to the following prompt:

```
Change Menu for the /etc/svc.conf file

        aliases             => 0
```

```
auth                => 1
group               => 2
hosts               => 3
netgroup            => 4
networks            => 5
passwd              => 6
protocols           => 7
rpc                 => 8
services            => 9

all of the above    => 10
none of the above   => 11

Enter your choice(s): 0 2
```

Press the Return key when you are through listing the databases.

If you opt not to edit any databases, the `svcsetup` command exits.

3. Displays a menu of the possible combinations of naming services that you can run on your system, with a number corresponding to each combination.

```
local               => 1
yp                  => 2
bind                => 3
local,yp            => 4
local,bind          => 5
yp,local            => 6
bind,local          => 7
```

After the `svcsetup` command lists the recommended naming service combinations, it prompts you to specify the naming service order you want for each of the databases you are changing. Enter the number that corresponds to the new naming service order that you want to run.

The default service order setting is displayed in brackets ([]).

For example, you chose to modify the service order for the `aliases` and `group` databases. The `svcsetup` command prompts you in the following manner:

```
Enter the naming service order for the "aliases" database [5]: 1

local               => 1
yp                  => 2
bind                => 3
local,yp            => 4
local,bind          => 5
yp,local            => 6
bind,local          => 7

Enter the naming service order for the "group" database [5]:

local               => 1
yp                  => 2
bind                => 3
local,yp            => 4
local,bind          => 5
yp,local            => 6
bind,local          => 7
```

The `[5]` indicates that the default entry in the `svc.conf` file. The preceding example shows how to change the current setting of the `aliases` database to `local`, and the setting of the `group` database to `local, bind`.

4. Prints an informational message to the screen that it is updating the `svc.conf` file, and exits.

Note

Resolving a name locally is always faster than using a name service. Therefore, as you add name services, such as YP and BIND, to your system, place the `local` service first in the `svc.conf` file.

You should always have the `local` service selected for the `passwd` and `hosts` database.

Only `local` and `bind` are valid for the `auth` database and only `yp` is valid for the `netgroup` database.

If you do not see the naming service order that you would like to use displayed in the menu, you must edit the `svc.conf` file with an editor, such as `vi`. The naming service orders displayed in the preceding examples are recommendations. All combinations work as you would expect. For example, you may want to use `local`, `yp`, `bind` for the `hosts` database.

3.1.2 Editing the svc.conf File Manually

You must be logged onto your system as root or superuser to edit the `svc.conf` file. Invoke an editor, and add or modify the database entries. The minimum number of `service` entries is one; the maximum number is three. Each entry should be on a new line and of the following form:

```
database=service,service,service
```

White space is permitted after commas, and after new lines, but is not required.

3.2 Using the bindsetup Command

You can use the `bindsetup` command to configure your system as a BIND/Hesiod client, primary server, secondary server, slave server, or caching server. You can also use it to remove the BIND/Hesiod service from your system.

Before you run `bindsetup` be sure that the following conditions exist:

- The system is in multiuser mode
- The network is up

3.2.1 Configuring a BIND/Hesiod Client

Before running the `bindsetup` command you must know your default domain name and the host names and addresses of the BIND/Hesiod servers you want to specify.

You can run `bindsetup` either noninteractively or interactively to configure a BIND/Hesiod client. To run it noninteractively supply the domain name and server names and IP addresses on the command line. For example, if the domain name is `cities.dec.com`, the server names are `orange` and `apple`, and the servers' IP addresses are `128.11.22.33` and `128.11.22.44` type the following:

```
# bindsetup -c -b cities.dec.com orange,128.11.22.33 apple,128.11.22.44
```

The bindsetup command then sets up your system silently as a BIND/Hesiod client with the domain and servers specified.

To run the bindsetup command interactively, type:

```
# bindsetup
```

The bindsetup command then steps through the setup procedure in the following manner:

1. Displays a menu asking you whether you want to add to or modify your BIND/Hesiod environment, remove BIND/Hesiod from your system, or exit bindsetup.

   Select the add/modify option.

2. Prompts you for a confirmation that you must know the default domain name to continue.

3. Prompts you for the default domain name, for example:

   ```
   cities.dec.com
   ```

   After you supply the default domain name, bindsetup displays a configuration menu that asks whether you want to configure your system as a primary, secondary, slave server, caching server, or as a client.

   Select the client option.

4. Prompts you for the host name and Internet address of the server, or servers, in your domain.

   Enter information for at least one server.

5. Lists the files that it is updating.

6. If the default domain name does not exist on the DD rule in the /etc/sendmail.cf file, binsdsetup will add it and remind you to kill the sendmail process, refreeze the sendmail.cf file, and restart sendmail.

7. Prints a message reminding you to edit the /etc/svc.conf file manually or by running the svcsetup command.

If your host name is not already set to the fully qualified domain name, the bindsetup command edits the /etc/hosts and /etc/rc.local files. For example, if your host name is chicago and your default domain name is cities.dec.com, your system's /etc/hosts entry will look like this:

```
128.11.22.33 chicago.cities.dec.com chicago
```

Edit your /etc/hosts file and add the default domain name to all host names by following the previous example. Your /bin/hostname line in /etc/rc.local will look like this:

```
/bin/hostname chicago.cities.dec.com
```

The bindsetup command also executes the /bin/hostname command. However, to ensure that your system knows its new host name, you should reboot your system. To do so, enter the following command:

```
# /etc/shutdown -r now
```

3.2.2 Configuring a BIND/Hesiod Primary Server

You can use the `bindsetup` command to set up your system as a BIND/Hesiod primary server.

Note

Before running `bindsetup`, copy the `etc` style files that you plan to distribute to the `/var/dss/namedb/src` directory on the primary server. The valid file names for the directory are `aliases`, `auth`, `group`, `hosts`, `networks`, `passwd`, `protocols`, `rpc`, and `services`.

Invoke the `bindsetup` command, specifying no options:

```
# bindsetup
```

The `bindsetup` command steps through the setup procedure in the following manner:

1. Displays a menu asking you whether you want to add to or modify your BIND environment, remove BIND from your system, or exit `bindsetup`.

   Select the `add/modify` option.

2. Prompts you for a confirmation that you must know the default domain name to continue.

3. Prompts you for the default domain name. After you supply the default domain name, `bindsetup` displays a configuration menu, which asks whether you want to configure your system as a primary, secondary, slave server, caching server, or as a client.

   Select the `primary` option.

4. Searches the `/var/dss/namedb/src` directory for `etc` style source files from which to create the BIND/Hesiod databases. It lists the files it finds, asking you if you want `bindsetup` to create the BIND/Hesiod databases from these source files.

   If you answer yes, `bindsetup` runs a series of scripts located in `/var/dss/namedb/bin` that convert `etc` style files to the appropriate BIND/Hesiod database format and places them in the `/var/dss/namedb` directory. The names of the BIND/Hesiod database files consist of the database name with a `.db` extension. When `bindsetup` creates the `hosts` database, it also creates the `hosts.rev` file. The `hosts.rev` file contains the Internet Protocol (IP) address to hostname mapping.

   If you answer no, or if the `/var/dss/namedb/src` directory is empty, `bindsetup` edits the appropriate system files with the entries that BIND/Hesiod requires to operate, but prints a warning message that you must create the BIND/Hesiod service database files once setup is complete.

   If you are serving the `passwd` database, `bindsetup` adds the Hesiod update daemon, `hesupd`, to the `/etc/rc.local` file. The `hesupd` daemon allows users to remotely change their password with the `passwd` command. The `hesupd` daemon requires the alias, `bindmaster`, on the primary server's `hosts` entry. The `bindsetup` command edits the `/var/dss/namedb/src/hosts` file to insert this alias. See Appendix A for more information.

The bindsetup command also updates the appropriate system files with the entries that BIND/Hesiod requires to operate.

Note

If your system was running ULTRIX-32, Version 3.0 or Version 3.1 software and you have been using BIND as your network host lookup service, your host information was located in the /etc/namedb directory. In order for bindsetup to create the BIND/Hesiod service hosts.db and hosts.rev database files, you must change the BIND database style file of host information to an etc style file, named hosts, and place it in the /var/dss/namedb/src directory.

5. Asks you if you want to run a Kerberos authenticated named daemon. If you select this option you must have already set up the appropriate Kerberos files. The bindsetup command prints an informational message about which files are required, and asks you if you have them set up. It also asks you if you already have a Kerberos master set up in your local area network, and what its name and IP address is.

   If you indicate that you would like to run a Kerberos authenticated named, but answer no to either question, bindsetup does not set up named with Kerberos options.

   See the following reference pages for information on Kerberos: acl_check(3krb), des_crypt(3krb), ext_srvtab(8krb), kinit(8krb), kdb_destroy(8), kdb_edit(8krb), kerberos(3), kerberos(3krb), klist(8krb), and krb.conf(5krb). You can also refer to the *Guide to Kerberos* for more information.

6. Creates the appropriate boot file for your BIND server. This file is /var/dss/namedb/named.boot by default.

7. Lists the files that it is updating.

8. If the bindsetup command created the BIND/Hesiod database files, it asks if you would like to start the named daemon atuomatically.

9. If the default domain name does not exist in the DD rule in the /etc/sendmail.cf file, bindsetup will add it and remind you to kill the sendmail process and refreeze the sendmail.cf file and restart sendmail.

10. Prints a message reminding you to edit the /etc/svc.conf file either manually or by running the svcsetup command.

If your host name is not already set to the fully qualified domain name, the bindsetup command edits the /etc/hosts and /etc/rc.local files. For example, if your host name is chicago and your default domain name is cities.dec.com, your system's /etc/hosts entry will look like this:

```
128.11.22.33 chicago.cities.dec.com chicago
```

Edit your /etc/hosts file and add the default domain name to all host names by following the previous example. Your /bin/hostname line in /etc/rc.local will look like this:

```
/bin/hostname chicago.cities.dec.com
```

The bindsetup command also executes the /bin/hostname command. However, to ensure that your system knows its new host name, you should reboot your system. To do so, enter the following command:

```
# /etc/shutdown -r now
```

If you need to know the process identification (pid) number of the daemon once it is running, check the file /etc/named.pid.

3.2.3 Configuring a BIND/Hesiod Secondary or Slave Server

You can use the bindsetup command to set up your system as a BIND/Hesiod secondary or slave server.

The setup is the same for both types of servers, although they function differently. For an explanation of the various servers' functions, see Chapter 1.

Invoke the bindsetup command, specifying no options:

```
# bindsetup
```

The bindsetup command steps through the setup procedure in the following manner:

1. Displays a menu asking you whether you want to add to or modify your BIND environment, remove BIND from your system, or exit bindsetup.

   Select the add/modify option.

2. Prompts you for a confirmation that you must know the default domain name to continue.

3. Prompts you for the default domain name. After you supply the default domain name, bindsetup displays a configuration menu, which asks whether you want to configure your system as a primary, secondary, slave server, caching server, or as a client.

   Select the secondary or slave option, depending on which you want to configure.

4. Prompts you for the host name and Internet address for the BIND/Hesiod primary server in your domain if you chose the secondary option.

5. Prompts you for one or more host names and Internet addresses of BIND/Hesiod servers in your domain to act as forwarders if you chose the slave option.

6. Asks you if you want to run a Kerberos authenticated named daemon. If you select this option, you must have already set up the appropriate Kerberos files. The bindsetup command prints an informational message about which files are required, and asks you if you have them set up. It also asks you if you already have a Kerberos master set up in your local area network, and what its name and IP address is.

   If you indicate that you would like to run a Kerberos authenticated named, but answer no to either question, bindsetup does not set up named with Kerberos options.

   See the following reference pages for information on Kerberos: acl_check(3krb), des_crypt(3krb), ext_srvtab(8krb), kinit(8krb), kdb_destroy(8krb), kdb_edit(8krb), kerberos(1), kerberos(3krb), klist(8krb), krb.conf(5krb), and krb.realms(5). You can also refer to

the *Guide to Kerberos* for more information.

7. Creates or updates the appropriate boot file for your BIND/Hesiod server. This file is `/var/dss/namedb/named.boot` by default.
8. Lists the files it is updating.
9. The `bindsetup` command asks if you would like to start the `named` daemon automatically.
10. If the default name does not exists in the DD rule in the `/etc/sendmail.cf` file, `bindsetup` will add it and remind you to `kill` the `sendmail` process and refreeze the `sendmail.cf` file either manually or by running the `svcsetup` command.
11. Prints a message reminding you to edit the `/etc/svc.conf` file manually or by running the `svcsetup` command.

If your host name is not already set to the fully qualified domain name, the `bindsetup` command edits the `/etc/hosts` and `/etc/rc.local` files. For example, if your host name is chicago and your default domain name is `cities.dec.com`, your system's `/etc/hosts` entry will look like this:

```
128.11.22.33 chicago.cities.dec.com chicago
```

Edit your `/etc/hosts` file and add the default domain name to all host names by following the previous example. Your `/bin/hostname` line in `/etc/rc.local` will look like this:

```
/bin/hostname chicago.cities.dec.com
```

The `bindsetup` command also executes the `/bin/hostname` command. However, to ensure that your system knows its new host name, you should reboot your system. To do so, enter the following command:

```
# /etc/shutdown -r now
```

If you need to know the process identification (pid) number of the daemon once it is running, check the file `/etc/named.pid`.

3.2.4 Configuring a BIND Caching Server

You can use the `bindsetup` command to set up your system as a BIND caching server.

Invoke the `bindsetup` command, specifying no options:

```
# bindsetup
```

The `bindsetup` command steps through the setup procedure in the following manner:

1. Displays a menu asking you whether you want to add to or modify your BIND environment, remove BIND from your system, or exit `bindsetup`.

   Select the `add/modify` option.
2. Prompts you for a confirmation that you must know the default domain name to continue.
3. Prompts you for the default domain name. After you supply the default domain name, `bindsetup` displays a configuration menu, which asks whether you

want to configure your system as a primary, secondary, slave server, caching server, or as a client.

Select the `caching` option.

4. Asks you if you want to run a Kerberos authenticated `named` daemon. If you select this option you must have already set up the appropriate Kerberos files. The `bindsetup` command prints an informational message about which files are required, and asks you if you have them set up. It also asks you if you already have a Kerberos master set up in your local area network, and what its name and IP address is.

   If you indicate that you would like to run a Kerberos authenticated `named`, but answer no to either question, `bindsetup` does not set up `named` with Kerberos options.

   See the following reference pages for information on Kerberos: `acl_check`(3krb), `des_crypt`(3krb), `ext_srvtab`(8krb), `kdb_destroy`(8krb), `kdb_edit`(8krb), `kerberos`(1), `kerberos`(3krb), `kinit`(8krb), `klist`(8krb), `krb.conf`(5krb), and `krb.realms`(5). You can also refer to the *Guide to Kerberos* for more information.

5. Creates or updates the appropriate boot file for your BIND/Hesiod server. This file is `/var/dss/namedb/named.boot` by default.
6. Lists the files it is updating.
7. Asks if you would like to start the `named` daemon automatically.
8. If the default domain name does not exist in the DD rule in the `/etc/sendmail.cf` file, `bindsetup` will add it and remind you to `kill` the `sendmail` process and refreeze the `sendmail.cf` file and restart `sendmail`.
9. Prints a message reminding you to edit the `/etc/svc.conf` file either manually or by running the `svcsetup` command.

If your host name is not already set to the fully qualified domain name, the `bindsetup` command edits the `/etc/hosts` and `/etc/rc.local` files. For example, if your host name is chicago and your default domain name is `cities.dec.com`, your system's `/etc/hosts` entry will look like this:

```
128.11.22.33 chicago.cities.dec.com chicago
```

Edit your `/etc/hosts` file and add the default domain name to all host names by following the previous example. Your `/bin/hostname` line in `/etc/rc.local` will look like this:

```
/bin/hostname chicago.cities.dec.com
```

The `bindsetup` command also executes the `/bin/hostname` command. However, to ensure that your system knows its new host name, you should reboot your system. To do so, enter the following command:

```
# /etc/shutdown -r now
```

If you need to know the process identification (pid) number of the daemon once it is running, check the file `/etc/named.pid`.

3.3 Removing the BIND/Hesiod Service

You can use the `bindsetup` command to remove the BIND/Hesiod service from your system. Before you remove it, however, you should edit the `/etc/svc.conf` file and remove all references to bind. After you have edited the file, invoke the `bindsetup` command specifying no options:

```
# bindsetup
```

The `bindsetup` command displays a menu asking you whether you want to add to or modify your BIND environment, remove BIND from your system, or exit `bindsetup`.

Select the `remove` option.

The `bindsetup` command issues a warning that BIND is not completely removed until you have removed all references to BIND from the `/etc/svc.conf` file.

Managing and Using the BIND/Hesiod Service 4

This chapter provides background information about the BIND/Hesiod service, and includes a description of roles of the domain administrator and the technical and zone contacts.

This chapter also describes how to register your site with the public networks and where to find additional information.

Finally, this chapter provides a brief tutorial on the commands you can use to obtain host names, IP addresses, and user information from the BIND/Hesiod service.

4.1 Maintaining the Domain

BIND/Hesiod domains are administrative entities that provide decentralized management of host names, addresses, and user information. The domain naming scheme is distributed and hierarchical. The Network Information Center (NIC) maintains the zone files of the root domain BIND server. The NIC also maintains the top-level domains `arpa` (gradually being phased out), `com`, `edu`, `gov`, `mil`, and `org`, plus a number of country domains. In addition, the NIC registers first and second-level domains.

The domain administrator (DA) administers each local domain with the help of the technical and zone contacts. These roles are described in the following sections.

4.1.1 Domain Administrator Role

Typically, each domain has a domain administrator (DA), who is responsible for coordinating and managing the domain. The DA registers a second-level or lower domain by interacting with the DA in the next higher level domain. For information on finding the names of the DA contacts, see Section 4.5.3.

The DA duties include:

- Understanding the concepts and procedures of the BIND/Hesiod service
- Ensuring that the service is reliable
- Ensuring that the BIND/Hesiod data is current
- Taking prompt action when necessary, for example if protocols are violated or other serious misbehavior occurs
- Controlling the assignments of the host and domain names

The DA furnishes users with access to names and name-related information both inside and outside the local domain. In addition, the DA works closely with the domain technical and zone contacts for the domain.

4.1.2 The Technical and Zone Contact

Typically, the technical and zone contact is concerned with the technical aspects of maintaining the BIND/Hesiod server and resolver software and the data files. The technical and zone contact keeps the BIND/Hesiod server running and interacts with technical people in other domains and zones to solve problems affecting the local domain.

A zone consists of those contiguous parts of the domain tree for which a domain server has complete information and over which it has authority. A BIND/Hesiod server can be the authority for several zones.

4.2 Naming Domains and Hosts

The NIC makes domain name assignments on a first-come, first-served basis. The NIC only registers domains under the top-level domains, not individual hosts. This allows administration and data maintenance to be delegated down the hierarchical tree.

A domain is identified by a domain name, and consists of that part of the domain name space that is at or below the domain name. A domain is a subdomain of another larger domain, if it is contained within that domain. That is, if a domain's name ends with the containing domain's name, of which it is a subdomain. For example, `A.B.C.D` is a subdomain of `B.C.D`, `C.D`, `D`, and the root domain (.).

There are two types of names:

- The fully qualified name represents the complete domain name. This is also known as the absolute or canonical name. For example:

  ```
  chicago.cities.dec.com.
  ```

- Relative names represent the starting name (label) of an absolute domain name. Relative names are incomplete, but are completed by the BIND service, using knowledge of the local domain, for example:

  ```
  chicago
  ```

  Relative host names such as `chicago` are automatically expanded to the fully qualified domain name (`chicago.cities.dec.com`) when given in a typical command.

Domain and host names must begin with a letter, end with a letter or digit, and have only letters, digits, or hyphens as internal characters. Although the names can be up to 64 characters, it is best to choose names that are 12 characters or fewer because the canonical (fully qualified) domain names are easier to keep track of if they are short. The sum of all the label octets and label lengths is limited to 255.

Note

Domain names are case insensitive. By convention, however, whenever you receive a domain name you should preserve its case.

It is up to the DA to resolve any local conflicts concerning the domain name chosen.

Countries can register as top-level domains provided they name themselves after a two-letter country code listed in the international standard ISO-3166. In the event that a country code is identical to a state code that the U.S. Postal Service uses, the country can request a three-letter code.

4.3 Registering With Public Networks

Before you can set up the BIND/Hesiod service on your system, your system must be established on a local area network. If the BIND/Hesiod service for your domain is part of a public network, contact the organization in charge of that network to request the appropriate domain registration form. Even if your site belongs to more than one network, you should register your site with only one. The following sections describe how to contact the following networks:

- DARPA Internet network (ARPANET)
- CSNET
- BITNET

4.3.1 Contacting the DARPA Internet Network

If your system is on the DARPA (Defense Advanced Research Projects Agency) Internet network (also known as the ARPANET), contact the following organization to obtain information about setting up a BIND domain:

```
hostmaster@nic.ddn.mil
```

You can also request to be placed on the BIND mailing list. This mailing list is for people running BIND on the DARPA Internet network who want to discuss future designs, operational problems, and other related topics. Here is the address:

```
bind-request@ucbarpa.berkeley.EDU
```

4.3.2 Contacting the CSNET

If your site's domain name is not already registered with the CSNET (Computer Science Network), contact the CSNET Coordination and Information Center (CIC). It will send you an application and provide you with information and technical advice about setting up a domain.

If your site's domain name is already registered with the CIC, keep the CIC informed of how you want your site's mail routed. In general, the CSNET relay prefers to send mail by CSNET, rather than by the BITNET or the ARPANET. If your site is on more than one network, the CSNET relay might not be the preferred route.

You can contact the CIC at the following electronic mail address:

```
cic@sh.cs.net
```

Or, you can reach the CIC hotline at this phone number:

(617) 873-2777

4.3.3 Contacting the BITNET

Some colleges and universities are on the BITNET network. This network is reserved for students, faculty, and scholars who want to communicate on a common network. BITNET stands for "Because It's Time Network."

If your site is on the BITNET and you want to set up a domain, contact the following address for information:

```
BITNET Network Information Center (BITNIC)
```

```
Educom
Bitnet Network Information Center
P.O. Box 364
Princeton, NJ 08540
(609) 520-3340
```

For general information, send electronic mail to:

```
bitserve@CUNYVM
```

For general inquiries, send electronic mail to:

```
info%bitnic.bitnet@CUNYVM.CUNY.EDU
```

4.4 Updating BIND/Hesiod Data Files on the Primary Server

Occasionally you may need to update the BIND/Hesiod data files. For example, you may need to add a host to the data files. To update the data file, for example to add a host, follow these steps:

1. Be sure the minimum refresh time in the SOA record for the secondary servers is 30 seconds.
2. Edit the `/var/dss/namedb/src/hosts` file to add the new host.
3. Change to the directory `/var/dss/namedb` and type the following:

   ```
   # make hosts
   ```

   or

   ```
   # make all
   ```

After you have edited the `hosts` file and issue the `make` command, the BIND/Hesiod conversion scripts, which are in the directory, `/var/dss/namedb/bin`, make the new `hosts` databases, `hosts.db`, and `hosts.rev` and place them in the directory, `/var/dss/namedb`, and send a signal to the `named` daemon to reload all databases that have changed.

The BIND/Hesiod database conversion scripts also increment the serial number field of the SOA entry in the database file. A sample `hosts.db` file is provided in Appendix B. When the secondary servers poll the primary server and see that the serial number field has changed, they know to refresh their data.

The process is the same for all of the valid files in the primary server's `/var/dss/namedb/src` directory.

Scripts are provided to create the following databases: `aliases.db`, `auth.db`, `group.db`, `hosts.db`, `hosts.rev`, `networks.db`, `passwd.db`, `protocols.db`, `rpc.db`, and `services.db`.

If you do not want to distribute all BIND/Hesiod databases, a primary server requires only the `hosts.db` and `hosts.rev` databases.

Note

If your primary server is serving `passwd.db`, you can run the `hesupd` daemon. The `hesupd` daemon allows users to change their Hesiod password remotely by using the `passwd` command.

4.5 Obtaining Host Name and IP Address Information

There are several ways that you can obtain information about host name, IP addresses, and user information from a system using the BIND/Hesiod service. The following sections provide an introduction to these commands:

- `nslookup`
- `nsquery`
- `whois`

4.5.1 The nslookup Command

One way to obtain information about host names, IP addresses, and user information is with the `nslookup` command. With this command, you can noninteractively and interactively query the BIND/Hesiod service for information about hosts on the local, as well as remote, domains. You can also find information about BIND resource records such as MX, NS, and so forth.

Here is the format for a noninteractive query with the `nslookup` command:

`nslookup` *hostname*

A good way to learn how to use the `nslookup` options is to experiment with it. Appendix C provides a sample interactive session with the `nslookup` command. For further information, see `nslookup`(1) in the ULTRIX Reference Pages.

To find out `MX` information, you need to supply the `nslookup` command with a bogus host name and a valid domain name. For example, to get an answer to the question, who takes mail for the domain `dec.com?` , you could type the following:

```
# nslookup
Default Server:  oops.cities.dec.com
Address:  128.54.54.1

> set querytype=mx
> findMX.dec.com.
Server:  oops.cities.dec.com
Address:  128.54.54.1

findMX.dec.com      preference = 100,
       mail exchanger = decwrl.dec.com
decwrl.dec.com        inet address = 128.54.54.79
decwrl.dec.com        inet address = 128.54.54.93
> <CTRL/d>
#
```

In this example, the host name `findMX.dec.com.` is bogus, but the domain `dec.com.` is real. See Appendix C for further examples of `nslookup` command sessions.

4.5.2 The nsquery Command

The `nsquery` command provides a quick, noninteractive method for obtaining host names, aliases, and IP addresses. The following example shows how to get the host name, alias, and IP address for a host called `chicago`:

```
# nsquery chicago
Name: chicago
Address: 128.11.22.333
Aliases: c ch
```

See nsquery(1)in for further information.

4.5.3 The NIC whois Service

The NIC whois service allows you to verify the following information:

- The name and address of the organization responsible for the domain
- The name of the domain
- The domain's administrative and technical and zone contacts
- The host names and network addresses of sites providing the BIND service for the domain

To use the NIC whois service to find information about a domain named rice.edu, send mail specifying the whois command and the domain in question in the subject header:

```
# mail service@nic.ddn.mil
Subject: whois domain rice.edu
CTRL/d
Cc:
Null message body; hope that's ok
```

Here is a sample response:

```
From SERVICE-REPLY@NIC.DDN.MIL Thu Jun  2 17:58:38 1988
Received: from chicago.cities.dec.com (chicago.ARPA) by paris.cities.dec.com (1.2/dv.5.yp)
        id AA17498; Thu, 2 Jun 88 17:57:20 edt
Received: by chicago.cities.dec.com (5.57/v2.4)
        id AA03640; Thu, 2 Jun 88 17:56:49 EDT
Message-Id: <8806022156.AA03640@chicago.cities.dec.com>
Date:  2-Jun-88 12:58:35
From: NIC Mail Service <SERVICE-REPLY@NIC.DDN.MIL>
To: jane@chicago (h jane ramburg-crane)
Subject: Re: whois domain rice.edu
Status: RO

  Rice California (RICE-DOM)
  Advanced Studies and Research
  Houston, TX  77001

  Domain Name: RICE.EDU

  Administrative Contact:
    Ramone, Ramon (KK28)  Ramone@LLL-CRG.ARPA (713) 555-4834
  Technical Contact, Zone Contact:
    Raffle, Win R.  (VVR)  raf@RICE.EDU
    (713)  555-8101  ext 3844

  Domain servers in listed order:

  RICE.EDU                      128.42.5.1
  PENDRAGON.CS.PURDUE.EDU       128.10.2.5
```

4.6 Obtaining Further Information about the BIND Service

The NIC has several online documents that you can access to obtain further information about the BIND service. Some of these documents are:

NETINFO:DOMAINS.TXT
: This file consists of an informational table of the top-level domains and their root servers. This file is updated each time a new top-level domain is approved.

NETINFO:DOMAIN-INFO.TXT
: This file contains a concise list of all top-level and second-level domain names registered with the NIC. This file is updated monthly.

NETINFO:DOMAIN-CONTACTS.TXT
: This file lists each of the top-level and second-level domains, and includes the administrative, technical and zone contacts for each as well.

NETINFO:DOMAIN-TEMPLATE.TXT
: This file contains the questionnaire to be completed before registering a top-level or second-level domain. A copy of this document is in Appendix B.

You can use the `ftp` command to transfer copies of the online documents from `NIC.DDN.MIL`. Appendix B provides a sample `ftp` session. Or, you can open a TCP or UDP connection to the NIC host name server, port 101 on NIC.DDN.MIL. From there, you can invoke the command `ALL-DOM`. Appendix C lists several other articles and RFCs that may be of interest to you.

For further information about the BIND service, do the following:

- Send electronic mail to:

  `HOSTMASTER@NIC.DDN.MIL`

- Call the toll-free NIC hotline at:

  `(800) 235-3155`

4.7 How to Make and Use a Hesiod Database

The Hesiod name service is designed to handle any text information. In ULTRIX, Version 4.0, Hesiod is being used to serve eight `/etc` style databases: `aliases`, `auth`, `group`, `networks`, `passwd`, `protocols`, `rpc`, and `services`.

However, you may design your own database and serve it using Hesiod. The following steps use the `/usr/lib/X11/rgb.txt` file as an example. Follow these steps:

1. Design an `/etc` style file and place it in the directory, `/var/dss/namedb/src`. The following is the contents of the example file, `colors`:

```
112 219 147     aquamarine
112 219 147     Aquamarine
50 204 153      medium aquamarine
50 204 153      MediumAquamarine
0 0 0           black
0 0 0           Black
95 159 159      cadet blue
95 159 159      CadetBlue
```

2. Write a script to parse the source file and build a Hesiod database. Use the script files in the directory, `/var/dss/namedb/bin`, as examples. Place the new script in that directory. The following is the output which is placed in the file, `colors.db` and is produced by the script, `make_colors`. The domain name for the example is `dec.com` and the server name is `server.dec.com`:

```
$origin colors.dec.com.
@       HS      SOA     server.dec.com. postmaster.server.dec.com. (
                                        1       ; Serial
                                        300     ; Refresh - 5 minutes
                                        60      ; Retry - 1 minute
                                        1209600; Expire - 2 weeks
                                        43200 ); Minimum - 12 hours
        HS      NS      server.dec.com.
;
; %COLORS_START% - entries added by /var/dss/namedb/bin/make_colors
Aquamarine              HS TXT          "112:219:147"
MediumAquamarine        HS TXT          "50:204:147"
medium_aquamarine       HS CNAME        MediumAquamarine
Black                   HS TXT          "0:0:0"
CadetBlue               HS TXT          "95:159:159"
cadet_blue              HS CNAME        CadetBlue
; %COLORS_END%
```

3. Modify the `Makefile` in the `/var/dss/namedb` directory to include the building of the new database. The lines needed in the `Makefile` to build the `colors` database are shown in the following example:

```
MAKE_CO=make_colors -v

all: ...  colors.db ...

colors.db: $(NAMEDSRCDIR)/colors
    -@if [ -f $(NAMEDSRCDIR)/colors ]; then
                $(NAMEDBINDIR)/$(MAKE_CO); \
                $(ECHO) "updated colors"; \
        else \
                $(ECHO) "couldn't find $(NAMEDSRCDIR)/colors"; \
        fi
 .
 .
 .
colors:         colors.db push
$(NAMEDSRCDIR)/colors:
```

4. Add new zone declaration lines to any primary or secondary server's `/var/dss/namedb/named.boot` file and restart the server. The following examples are zone declaration lines:

```
primary         colors.dec.com          colors.db

OR

secondary       colors.dec.com          128.11.22.33     colors.db
```

5. Write an application that uses a `hes_resolve()` call to get at the Hesiod data stored in the servers. The following example shows a section from an example X server that first tries to find the screen color in Hesiod, and if that fails, looks in the local `rgb` DBM files:

```
#include <hesiod.h>
char *screen_color = "CadetBlue";
```

```
char **color_string;
int color_number;

color_string = hes_resolve(screen_color, "colors");
if (color_string == NULL) {
     /*
      * do the dbm lookup locally
      */
     .
     .
     .
else {
     /*
      * parse the color_string in format, #:#:#
      */

     if (*color_string != NULL)
               color_number = process_string(color_string);
```

After step 4, try either dumping the server's cache and examining the cache dump file in `/var/tmp/named_dump.db` or using the `nslookup` command to make sure that the data is properly served by the servers. Once the application is written, you can begin using Hesiod to serve your own database.

Troubleshooting the BIND/Hesiod Service 5

This chapter contains guidelines for troubleshooting the BIND/Hesiod service as well as information for starting, controlling and debugging the `named` daemon.

If the BIND/Hesiod service fails to work properly, the cause is typically one of the following:

- The data files are not set up properly
- The BIND/Hesiod service cannot access the root servers

The following files and daemon are crucial to the proper working of the BIND/Hesiod service:

- The primary server data files are located in the directory `/var/dss/namedb` and are `named.boot`, `hosts.rev`, `hosts.db` `named.local`, and `named.ca`. Other `.db` files may exist if Hesiod is used.
- The `/etc/svc.conf` file
- The `/etc/rc.local` file
- The `/etc/resolv.conf` file
- The `/etc/hesiod.conf` file
- The `named` daemon (for BIND/Hesiod servers, only)
- The `named-xfer` daemon (for BIND/Hesiod secondary servers only)

The following sections describe these files and the daemons, located in the `/usr/etc`, directory in greater detail.

5.1 Reviewing the Domain Data Files

This section suggests several possible problem areas if the BIND/Hesiod service is not working properly.

First, be sure that the domain data files are set up correctly. Specifically, be sure that the following are correct:

- The `local host` entry in the boot file and cache files
- The `in-addr` domain in the boot file and any other database files
- The reverse `arp` IP addresses
- The host names are in the correct domain

In addition, be sure that there is only one reverse address for each host in the domain.

If the preceding information is correct and you are still having problems, continue troubleshooting the BIND/Hesiod service as described in the rest of this chapter.

For information about the domain data files, see Chapter 2. For examples of domain data files, see Appendix B.

5.2 Reviewing the /etc/rc.local File

Check to make sure that `/etc/rc.local` file contains the following entry:

```
# BINDSTART
echo -n 'BIND daemon:'    > /dev/console
[ -f /usr/etc/named ] && {
        /usr/etc/named /var/dss/namedb/named.boot
        echo -n ' named' > /dev/console
}
echo '.'
# BINDEND
```

If you are running a Kerberos authenticated `named` daemon, the `/etc/rc.local` file should contain the following entry:

```
# BINDSTART
echo -n 'BIND daemon:'    > /dev/console
[ -f /usr/etc/named ] && {
        /usr/etc/named -n -a kerberos.one -b /var/dss/namedb/named.boot
        echo -n ' named' > /dev/console
}
echo '.'
# BINDEND
```

The entry starts the domain name server each time the system goes to multiuser mode. It can be either before or after any YP entries, but should be before any NFS entries, if they exist. If neither YP nor NFS entries exist in the `/etc/rc.local` file, the `named` entry belongs before the local daemons, such as `sendmail`.

Note

Do not run the `named` daemon directly from `inetd`. This causes continual restarts of the name server and defeats the purpose of having a cache.

5.3 Reviewing the Resolver File

Make sure that the resolver file `/etc/resolv.conf` is accurate. It should contain the domain entry on all hosts running the BIND/Hesiod service, and a nameserver entry for `localhost` on servers. A client must also have at least one other nameserver entry.

See Appendixes A and B for information about the resolver file.

5.4 Reviewing the Hesiod File

Check the Hesiod file, `/etc/hesiod.conf`, to verify that it is correct. The right hand side (RHS) should be set to a dot (.), followed by the default domain name. The left hand side (LHS) should be NULL.

5.5 Reviewing the Debug Files

If after reviewing the `rc.local` file and the resolver file you are still having problems, there are several other files to help you troubleshoot the BIND/Hesiod service further. These files are:

- `/var/spool/mqueue/syslog`

- /var/tmp/named_dump.db
- /var/tmp/named.run
- /var/tmp/named.stats

The following section provides general information about the debug files and explains how to use them to troubleshoot the BIND/Hesiod service.

5.5.1 The syslog File

If the BIND service cannot access the root servers, it cannot resolve queries about hosts in other domains. One way to determine if the root servers are accessible is to look in the /var/spool/mqueue/syslog file. The key phrase is:

```
root hints too low
```

This key phrase indicates how many of the available root servers are accessible to your system. The minimum threshold is two, and the maximum is the number of root servers available at their various addresses (currently 10). If the number of root hints is too low, either the BIND files are not configured properly or one or more of the links to the root servers is down.

In addition, the named daemon may log error messages in the syslog file.

Here is a sample syslog file:

```
Jun 21 04:05:05 syslog restart
Jun 21 12:09:51 localhost: 1688 named: restarted
Jun 21 12:09:51 localhost: 1688 named: /var/dss/namedb/nammed.boot: No such file or directory
Jun 21 12:10:49 localhost: 1692 named: restarted
Jun 21 12:12:16 localhost: 1692 named: ..(new) named started..
Jun 21 12:17:30 localhost: 1705 sendmail: AA01705: message-id=<8806211616.AA01705@chicago.cities.dec.com>
Jun 21 12:17:31 localhost: 1705 sendmail: AA01705: from=jane, size=243562, class=0
Jun 21 12:17:49 localhost: 1707 sendmail: AA01705: to=jane@orlando, delay=00:01:18, stat=Sent
Jun 21 14:50:37 localhost: 1692 named: reloading nameserver
Jun 21 14:50:45 localhost: 1692 named: 0 root hints... (too low)
Jun 21 15:20:46 localhost: 1692 named: 0 root hints... (too low)
Jun 21 15:50:46 localhost: 1692 named: 0 root hints... (too low)
Jun 21 15:59:02 localhost: 1840 sendmail: AA01840: message-id=<8806211958.AA01840@chicago.cities.dec.com>
Jun 21 15:59:02 localhost: 1840 sendmail: AA01840: from=jane, size=835, class=0
Jun 21 15:59:12 localhost: 1842 sendmail: AA01840: to=jane@tempe, delay=00:00:20, stat=Sent
```

5.5.2 The named_dump.db File

If you send the named daemon a signal to dump the database, a copy of the database is dumped in the file /var/tmp/named_dump.db. Here is how to send the signal:

```
# kill -INT `cat /etc/named.pid`
```

By examining the resulting named_dump.db file, you can determine whether any of the BIND/Hesiod data files are set up incorrectly. Here are some things to look for:

- Is the local loopback correct?
- Is the in-addr entry correct?
- Is the localhost set up correctly?
- Are the reverse arp IP addresses correct?

- Is there a reverse address for each host?
- Are the host names in the correct domain?

5.5.3 The named.run File

If you turn on debugging, the results are logged in the file `/var/tmp/named.run`. There are up to 11 debug levels. Typically, however, you should debug the `named` daemon at level five. Then, from looking at the `named.run` file you should be able to get an idea of whether the BIND/Hesiod service is working properly. For example, the following line in the `named.run` file indicates a connection to a root server:

```
TCP connected
```

The following list describes what you might see in the `named.run` file to indicate a system that is running poorly:

- Several QUESTIONS, but no ANSWERS
- Several iterations of `findns`, which attempt to find a name server
- Several iterations of `schedretry`, which schedule another attempt to access a root server

Appendix B lists two `named.run` files: one for a system that is running the BIND service properly, the other for a system that is not.

5.5.4 The named.stats File

The `/var/tmp/named.stats` file lists the statistics for the BIND/Hesiod service. From this file you can see how much activity is being generated for the BIND/Hesiod server. To generate this file, send a signal to the `named` daemon. For example:

```
# kill -IOT `cat /etc/named.pid`
```

For more information about how to send signals to the `named` daemon see Section 5.6.

After the `named` daemon is running with the `-IOT` signal, it generates the `named.stats` file. Here is an example of this file:

```
###  Thu June 21 15:05:09 1988
3389 time since boot (secs)
3389 time since reset (secs)
72   input packets
72   output packets
69   queries
0    iqueries
0    duplicate queries
3    responses
0    duplicate responses
69   OK answers
0    FAIL answers
0    FORMERR answers
2    system queries
1    prime cache calls
1    check_ns calls
0    bad responses dropped
0    martian responses
0    Unknown query types
60   A querys
4    NS querys
```

```
300   SOA querys
3     PTR querys
2     MX querys
6     TXT querys
5     AXFR querys
3     ANY querys
```

The named.stats file may have entries for martian responses. A martian response indicates a query response from a host that is unknown to the server.

5.6 Obtaining the named Process ID

When the named daemon starts running, it places its process identification number (pid) in the file /etc/named.pid. This feature is useful for programs that need to send signals to the named daemon.

You can also find the named process ID (pid) by using the ps command.

5.7 Sending Signals to the named Daemon

You can send several signals to the running named process without having to restart it.

The signals you can send to named are as follows:

SIGHUP — This signal causes the named process to read the boot file and reload the databases. However, all previously cached data is lost. This is useful if you have made a change to a data file and you want named's internal database to reflect the change.

SIGXFSZ — This signal causes the named process to reload only the databases that have changed.

SIGINT — This signal dumps the current database and cache to the file /var/tmp/named_dump.db. This can give you an indication of whether the database was loaded correctly.

SIGUSR1 — This signal turns on debugging. Each subsequent USR1 signal increments the debug level. A good rule of thumb is to increment the debug level to five (this is accomplished by issuing the signal five times). The output is appended to the file /var/tmp/named.run.

After turning on debugging, you can use the nslookup command to watch the debug trace. Appendix B has an example of two named.run files; one is from a system with the BIND/Hesiod service running properly, the other is from a system that cannot reach any of the root servers.

Here is an example of how to send the USR1 signal to the named daemon:

```
# kill -USR1 `cat /etc/named.pid`
```

You can start the named daemon in debug mode by typing the following command:

```
# /usr/etc/named /var/dss/namedb/named.boot -d5 &
```

This command starts the named daemon and sets the debug level to five.

SIGUSR2 — This signal turns off debugging completely.

```
# kill -USR2 `cat /etc/named.pid`
```

SIGKILL	This signal terminates the `named` process. Also, to stop the library routines from querying BIND, remove the `bind` (or `BIND`) entry in the `/etc/svc.conf` file.
SIGIOT	This signal dumps the BIND/Hesiod statistics for the file, `/var/tmp/named.stats`.

Configuring the BIND/Hesiod Service Manually A

This appendix explains how to configure your system manually as a BIND/Hesiod client or server. To configure the BIND/Hesiod service manually you must create or edit the appropriate files. If your system is a server you must be certain that the `named` daemon is running correctly.

The following sections describe:

- Configuring a BIND/Hesiod Client
- Configuring a BIND/Hesiod Server

For information on using the `svc.conf` file, see Chapter 3.

A.1 Configuring a BIND/Hesiod Client

To set up your system manually as a BIND client, do the following:

1. Edit the `/etc/svc.conf` file (See Chapter 3)
2. Create the file `/etc/resolv.conf`
3. Create the file `/etc/hesiod.conf`
4. Edit the `/etc/hosts` file

A.1.1 Create the resolv.conf File

The resolver file, `/etc/resolv.conf`, designates the BIND/Hesiod servers on the network.

The `resolv.conf` file must exist for all systems running the BIND/Hesiod service. The minimum it must contain is the domain entry and a nameserver entry. A client must have nameserver entries which tell the resolver the server IP address to use to answer a query. It is best to have additional nameserver entries, one for each additional server. Server replication reduces the possibility of the BIND/Hesiod service being interrupted in the event that a server goes down.

The `/etc/resolv.conf` file entries have the following format:

```
domain          domainname
nameserver      IP address
nameserver      IP address
```

See Appendix B for a sample `resolv.conf` file and see `resolv.conf`(5) in the *ULTRIX Reference Pages* for more information about the `/etc/resolv.conf` file.

A.1.2 Create the hesiod.conf File

The `hesiod.conf` file lists the `rhs` and `lhs` extensions. The `rhs`, for ULTRIX, is a dot (.) , followed by the domain name. The `lhs` is blank. See Appendix B.1.1 for a sample `hesiod.conf` file.

A.1.3 Edit the hosts File

You must add the names and IP addresses of all BIND servers that you in the setup files to the `/etc/hosts` file.

A.2 Configuring a BIND/Hesiod Server

BIND/Hesiod servers make use of several configuration files. To set up your system as a BIND/Hesiod server, you must first set up the BIND/Hesiod environment by editing these configuration files. After you have set up the BIND/Hesiod environment, you must start the `named` daemon.

Do the following to set up your system as a BIND/Hesiod server:

1. Edit `/etc/svc.conf` file (See Chapter 3)
2. Create `/etc/resolv.conf` file
3. Create the `/etc/hesiod.conf` file
4. Edit the `/etc/hosts` file
5. Edit the boot file
6. Create the database files (primary server only)
7. Edit the Domain Data Files
8. Edit the `/etc/rc.local` file
9. Start the `named` daemon

For an explanation of steps 1 through 4, see Appendix A.1.1 through A.2.3.

A.2.1 Edit the Boot File

When the `named` daemon starts running, it reads the boot file. This file tells the server what type of server it is, which zone it has authority for, and where to get its initial data. The boot file has the following default path and name:
`/var/dss/namedb/named.boot`

The boot file `named.boot` has two kinds of entries, one that specifies the directory where the data files that the `named` daemon reads at start time reside, and another that specifies the type of server. The format for each is as follows:

directory *directory*
type domain/zone source data file/IP addr

The `directory` entry allows you to state just the file name in subsequent entries in the boot file. If you do not have a `directory` entry in the boot file, you must explicitly state the full pathname for each file name in each entry.

The default directory is `/var/dss/namedb`. If you change the default directory, be sure that the directory you specify is large enough to hold a core dump, should one occur. This is especially important if there are include files with relative path

names called by `$include`. See Chapter 2 for information about the `include` data file entry.

The first field of the type entry specifies the type of server. The choices are: primary, secondary, forwarder, or slave. The server type determines how to interpret the information in the remaining fields.

The second field specifies the domain or zone for which the server has authority. The third field specifies the name of the file from which the data is to be read. The fourth field, which is optional, specifies the interval, in seconds, for refreshing the data file. The exact content of each field depends upon the type of server. See Appendix B for the boot file formats for the different servers.

A.2.2 Edit the Domain Data Files

The following are the four standard files for specifying the data for a domain: `named.ca`, `named.local`, `hosts.db` and `hosts.rev`.

The boot file, which is usually `/var/dss/named/named.boot`, specifies the location of these files. A description of these files follows:

- named.ca

  This file identifies the authoritative server for the zone. The `named.ca` file contains the necessary entries for the root servers.

  The format of the `named.ca` file follows the standard described in Chapter 2.

- named.local

  This file specifies the address for the local loopback interface, and is typically expressed as `localhost` with the network address 127.0.0.1.

- hosts.db

  This database file contains the host and address information for all the systems in the zone. It is created from the `/var/dss/namedb/src` file using the `make_hosts` script, located in the `/var/dss/namedb/bin` directory.

- hosts.rev

  This file specifies the `in-addr.arpa` domain, which allows reverse address to name mapping. This special domain was formed to allow inverse mapping because IP host addresses do not fall within domain boundaries.

  The `in-addr.arpa` domain has four labels preceding it, which correspond to the four octets of an IP address. You must specify all four octets even if an octet is zero. For example, the IP address 128.32.0.4 is located in the domain 4.0.32.128.in-addr.arpa. This address reversal is awkward to read but allows for the natural grouping of hosts in a network. This file is also created by the `make_hosts` script.

Examples of these files are shown in Appendix B. See the *Guide to Networking* for further information about IP addresses.

A.2.3 Edit the rc.local file

To have the BIND/Hesiod service start automatically on the BIND/Hesiod servers each time the system is brought to multiuser mode, place an entry for the `named` daemon in the `/etc/rc.local` file. This entry should go before any local daemon entries, such as `sendmail`. The entry for the `named` daemon can go either

before or after the entries for YP, but should go before any NFS entries, if they exist. The following is a typical entry:

```
# %BINDSTART% - BIND daemon

echo -n 'BIND daemon:'    > /dev/console
[ -f /usr/etc/named ] && {
     /usr/etc/named /var/dss/namedb/named.boot
        echo -n ' named'  >/dev/console
}
echo '.'      > /dev/console
# %BINDEND%
```

If your system is running a Kerberos Authenticated `named`, the `rc.local` entry should read as follows:

```
# %BINDSTART% - BIND daemon

echo -n 'BIND daemon:'    > /dev/console
[ -f /usr/etc/named ] && {
     /usr/etc/named -n -a kerberos.one -b /var/dss/namedb/named.boot
        echo -n ' named'  >/dev/console
}
echo '.'      > /dev/console
# %BINDEND%
```

Note that to run a Kerberos Authenticated `named` daemon requires the presence of certain system files, and a specific network configuration. Be certain that your system and network are correctly configured before starting `named` with Kerberos Authentication.

See the following reference pages for information on Kerberos: `acl_check`(3), `des_crypt`(3), `ext_srvtab`(8), `kdb_destroy`(8), `kdb_edit`(8), `kerberos`(1), `kerberos`(3), `kinit`(1), `klist`(1), `krb.conf`(5), and `krb.realms`(5). You can also see the *Guide to Kerberos*.

If you are setting up a primary server and are serving the `passwd` database, be sure to add an entry for the `hesupd` daemon. Enter the following lines before # `%BINDEND%`:

```
[ -f /usr/etc/hesupd ] && {
     /usr/etc/hesupd; echo -n ' hesupd'  >/dev/console
}
```

The `hesupd` daemon allows users to change their passwords from a remote system using the `passwd` command.

The lines beginning with a number sign (#) are comments that make the `rc.local` file easier to read.

Note

To maximize your system's performance when you are running BIND/Hesiod include only the following in the `/etc/hosts` file:

- The name of the local loopback, `localhost`.
- The name of your system.
- The name of any BIND/Hesiod servers on your local network.
- The name of any Kerberos servers, if they exist and Kerberos is being used.

Include a few other hosts in the `/etc/hosts` file, such as those needed in single-user mode.

A.2.4 Start the named daemon

To start the named daemon type the following command:

```
# /usr/etc/named /var/dss/namedb/named.boot
```

When the named daemon is started the BIND/Hesiod service starts automatically.

You can also reboot your system to start the named daemon, if you prefer. The following command shuts down the system and reboots it immediately:

```
# /etc/shutdown -r now
```

See shutdown(8) in the *ULTRIX Reference Pages* for further information about shutting down the system.

Sample Files B

This appendix includes examples of configuration and data files that the BIND/Hesiod service requires to resolve queries about hosts on the Internet, and about about network objects. With the exception of the following files, all files in this appendix are created by the `bindsetup` command: `svc.conf` , `/var/tmp/named.run` , and `/var/tmp/named_dump.db`.

The following configuration files should be located in the `/etc` directory:

- `hesiod.conf`

  Defines the expansion rules for the LHS and RHS of the *HesiodName* string

- `resolv.conf`

  Defines the name-value pairs that provide resolver information

- `svc.conf`

  Defines the order in which to query the name services that are running on a system

Note

The `/etc/svc.conf` file is a mandatory system file. Do not remove or move it.

The following data files are located in the BIND/Hesiod server data file directory, `/var/dss/namedb`. The files provide information about the network, network configuration, and network objects that BIND and Hesiod require to resolve queries.

- `named.boot` -- BIND server boot file
- `named.ca` -- BIND server cache file
- `named.local` -- BIND server local host reverse address host file

Note

Sample BIND files are not provided for root servers. If you are establishing your system as a root server, you can get help from the NIC, as stated in Chapter 1.

- `aliases.db` -- aliases database file
- `auth.db` -- authorization database file
- `group.db` -- group information database file
- `hosts.db` -- hosts database file
- `hosts.rev` -- reverse address hosts database file

- networks.db -- network information database file
- passwd.db -- password database file
- protocols.db -- network protocols database file
- rpc.db -- rpc (remote procedure call) database file
- services.db -- system services database file

When debugging is turned on for the named daemon the results are recorded in /var/tmp/named.run.

When the named daemon is sent a SIGINT signal, a dump of named daemon's cache is created in the /var/tmp/named_dump.db.

B.1 Configuration Files

The following sections provide samples of configuration files located in the /etc directory:

B.1.1 Sample hesiod.conf file

The following is a sample hesiod.conf file:

```
;RHS table
rhs=.cities.dec.com
;LHS table
lhs=
```

In this example, the RHS of a Hesiod query is expanded to .cities.dec.com.

B.1.2 Sample resolv.conf file

The following is a sample resolv.conf file for a server:

```
;
; BIND data file.
;
domain          cities.dec.com
nameserver      127.0.0.1
```

The following is a sample resolv.conf file for a client:

```
;
; BIND data file
;
domain          cities.dec.com
nameserver      128.11.22.33
nameserver      128.11.22.44
```

B.1.3 Sample svc.conf file

The following is a sample svc.conf file. You must have this file in the /etc directory:

```
# @(#)svc.conf  2.5     (ULTRIX)        8/15/89
#
# WARNING: This file is MANDATORY !
#
```

```
# Setup recommendation: As you add distributed services to database
#       entries, it is recommended that "local" is the first service.
#       For example:
#                       passwd=local,bind
#
# Note: White space allowed only after commas or newlines.
#
# File Format
# -----------
# database=service,service
#
# The database can be:
#       aliases
#       auth
#       group
#       hosts
#       netgroup
#       networks
#       passwd
#       protocols
#       rpc
#       services
# The service can be:
#       local
#       yp
#       bind
#
#
aliases=local
auth=local
group=local
hosts=local
netgroup=local
networks=local
passwd=local
protocols=local
rpc=local
services=local


PASSLENMIN=6
PASSLENMAX=16
SOFTEXP=604800          # 7 days in seconds
SECLEVEL=BSD            # (BSD | UPGRADE | ENHANCED)
```

B.2 Sample named.boot File

Only BIND servers need a boot file. The default name and location of the boot file is `/var/dss/namedb/named.boot`. This section provides a sample boot file for each of the following BIND servers: a primary master, a secondary master, a slave, and a caching. Note that each type of server needs an entry of the form:

```
primary  0.0.127-in-addr.arpa   named.local
```

This entry provides the address-to-hostname translation for the `localhost`.

B.2.1 Sample Primary Server Boot File

The following is a sample primary server boot file:

```
; Data file to boot a BIND primary server.
;
```

```
; directory where all the data files are stored
directory       /var/dss/namedb
;
; type          domain                          source host/file
primary         cities.dec.com                  hosts.db
;
primary         22.11.128.in-addr.arpa          hosts.rev
;
primary         aliases.cities.dec.com          aliases.db
primary         group.cities.dec.com            group.db
primary         networks.cities.dec.com         networks.db
primary         passwd.cities.dec.com           passwd.db
primary         protocols.cities.dec.com        protocols.db
primary         rpc.cities.dec.com              rpc.db
primary         services.cities.dec.com         services.db
;
primary         0.0.127.in-addr.arpa            named.local
;
; load the cache data last
cache                                           named.ca
```

B.2.2 Sample Secondary Server Boot File

The following is a sample secondary server boot file:

```
; Data file to boot a BIND secondary server.
;
; directory where all the data files are stored
directory    /var/dss/namedb
;
; type       domain                    source host/file
secondary    cities.dec.com            128.11.22.33  hosts.db
;
secondary    33.22.128.in-addr.arpa    128.11.22.33  hosts.rev
;
secondary    aliases.cities.dec.com    128.11.22.33  aliases.db
secondary    group.cities.dec.com      128.11.22.33  group.db
secondary    networks.cities.dec.com   128.11.22.33  networks.db
secondary    passwd.cities.dec.com     128.11.22.33  passwd.db
secondary    protocols.cities.dec.com  128.11.22.33  protocols.db
secondary    rpc.cities.dec.com        128.11.22.33  rpc.db
secondary    services.cities.dec.com   128.11.22.33  services.db
;
primary      0.0.127.in-addr.arpa      named.local
;
; load the cache data last
cache                                  named.ca
```

B.2.3 Sample Slave Server Boot File

The following is a sample slave server boot file:

```
; Data file to boot a BIND slave server.
;
; directory where all the data files are stored
directory         /var/dss/namedb
;
; type            domain                  source host/file
primary           0.0.127.in-addr.arpa    named.local
;
slave
forwarders        128.11.22.33  128.11.22.33
```

B.2.4 Sample Caching Server Boot File

The following is a sample caching server boot file:

```
;
; BIND data file to boot a caching only name server.
;
; directory where all the data files are stored
directory       /var/dss/namedb
;
; type          domain                  source host/file
primary         0.0.127.in-addr.arpa    named.local
;
; load the cache data last
cache                                   named.ca
```

B.3 Sample named.ca File

The following is a sample `named.ca` file: Only BIND servers running the `named` daemon need a cache file. The default name and location of the cache file is `/var/dss/namedb/named.ca`. Here is a sample cache file:

```
;
; BIND data file for initial cache data for root domain servers.
;
.               99999999        IN      NS      nic.ddn.mil.
.               99999999        IN      NS      ns.nasa.gov.
.               99999999        IN      NS      terp.umd.edu.
.               99999999        IN      NS      a.isi.edu.
.               99999999        IN      NS      aos.brl.mil.
.               99999999        IN      NS      gunter-adam.af.mil.
.               99999999        IN      NS      c.nyser.net.
nic.ddn.mil.    99999999        IN      A       26.0.0.73       ; Jeeves
                99999999        IN      A       10.0.0.51
ns.nasa.gov.    99999999        IN      A       192.52.195.10   ; BIND
a.isi.edu.      99999999        IN      A       26.3.0.103      ; Jeeves
                99999999        IN      A       128.9.0.107
aos.brl.mil.    99999999        IN      A       192.5.25.82     ; BIND
                99999999        IN      A       128.20.1.2
gunter-adam.af.mil. 99999999    IN      A       26.1.0.13       ; Jeeves
c.nyser.net.    99999999        IN      A       192.33.4.12     ; BIND
terp.umd.edu.   99999999        IN      A       128.8.10.90     ; BIND
```

B.4 Sample named.local File

Only BIND servers need a local file. The default name and location of the local file is `/var/dss/namedb/named.local`. Here is a sample `named.local` file:

```
;
; BIND data file for local loopback interface.
;
@       IN      SOA     chicago.cities.dec.com. postmaster.chicago.cities.dec.com. (
                        1       ; Serial
                        3600    ; Refresh
                        300     ; Retry
                        3600000 ; Expire
                        3600 )  ; Minimum
        IN      NS      chicago.cities.dec.com.
1       IN      PTR     localhost.
localhost.IN    A       127.0.0.1
```

B.5 Sample aliases.db File

The following is a sample `aliases.db` file:

```
$origin aliases.cities.dec.com.
@    HS    SOA     chicago.cities.dec.com. postmaster.chicago.cities.dec.com. (
                   15      ; Serial
                   300     ; Refresh - 5 minutes
                   60      ; Retry - 1 minute
                   1209600 ; Expire - 2 weeks
                   43200 ) ; Minimum - 12 hours
     HS    NS      chicago.cities.dec.com.
;
; %ALIASES_START% - entries added by /var/dss/namedb/bin/make_aliases
;
projectX      HS      TXT     "smith,hartwell,culligan,harris"
pizza_group   HS      TXT     "bob,sam,sue,jim,jane"
sysmgrs       HS      TXT     "tom,todd,greg,jill"
; %ALIASES_END%
```

B.6 Sample auth.db File

The following is a sample `auth.db` file:

```
$origin auth.cities.dec.com.
@    HS    SOA  chicago.cities.dec.com. postmaster.chicago.cities.dec.com. (
                13      ; Serial
                300     ; Refresh - 5 minutes
                60      ; Retry - 1 minute
                1209600 ; Expire - 2 weeks
                43200 ) ; Minimum - 12 hours
     HS    NS   chicago.cities.dec.com.
;
; %AUTH_START% - entries added by /var/dss/namedb/bin/make_auth
1            HS    TXT    "1:Nologin:616968156:360:5184000:07:0:1:00:00:00"
"auth-0"     HS    CNAME  1
2            HS    TXT    "2:Nologin:616968156:360:5184000:07:0:2:00:00:00"
"auth-1"     HS    CNAME  2
3            HS    TXT    "3:Nologin:616968156:360:5184000:07:0:3:00:00:00"
"auth-2"     HS    CNAME  3
; %AUTH_END%
```

B.7 Sample group.db File

The following is a sample `group.db` file:

```
$origin group.cities.dec.com.
@    HS    SOA   chicago.cities.dec.com. postmaster.chicago.cities.dec.com. (
                 13        ; Serial
                 300       ; Refresh - 5 minutes
                 60        ; Retry - 1 minute
                 1209600   ; Expire - 2 weeks
                 43200 )   ; Minimum - 12 hours
     HS    NS    chicago.cities.dec.com.
;
; %GROUP_START% - entries added by /var/dss/namedb/bin/make_group
system          HS      TXT     "system:*:0:root"
"0"             HS      CNAME   system
"group-0"       HS      CNAME   system
daemon          HS      TXT     "daemon:*:1:daemon"
"1"             HS      CNAME   daemon
"group-1"       HS      CNAME   daemon
```

```
uucp            HS      TXT     "uucp:*:2:uucp"
"2"             HS      CNAME   uucp
"group-2"       HS      CNAME   uucp
; %GROUP_END%
```

B.8 Sample hosts.db File

The following is a sample `hosts.db` file:

```
;
; Data file of hostnames in this zone.
;
@  IN   SOA   chicago.cities.dec.com. postmaster.chicago.cities.dec.com. (
              1       ; Serial
              300     ; Refresh - 5 minutes
              60      ; Retry - 1 minute
              1209600 ; Expire - 2 weeks
              43200 ) ; Minimum - 12 hours
   IN   NS    chicago.cities.dec.com.
;
; %HOSTS_START% - entries added by /var/dss/namedb/bin/make_hosts
localhost       IN      A               127.0.0.1
"hosts-0"       IN      CNAME           localhost
chicago         IN      A               128.11.22.33
"hosts-1"       IN      CNAME           chicago
chi             IN      CNAME           chicago
bindmaster      IN      CNAME           chicago
boston          IN      A               128.11.22.44
"hosts-2"       IN      CNAME           boston
bo              IN      CNAME           boston
newyork         IN      A               128.11.22.55
"hosts-3"       IN      CNAME           newyork
ny              IN      CNAME           newyork
; %HOSTS_END%
```

B.9 Sample hosts.rev File

The following is a sample `hosts.rev` file:

```
;
; Data file for reverse address to hostname.
;
@  IN   SOA   chicago.cities.dec.com. postmaster.chicago.cities.dec.com. (
              1       ; Serial
              300     ; Refresh - 5 minutes
              60      ; Retry - 1 minute
              1209600 ; Expire - 2 weeks
              43200 ) ; Minimum - 12 hours
   IN   NS    chicago.cities.dec.com.
;
; %HOSTS_START% - entries added by /var/dss/namedb/bin/make_hosts
33.22.11.128.in-addr.arpa.      IN      PTR     chicago.cities.dec.com.
44.22.11.128.in-addr.arpa.      IN      PTR     boston.cities.dec.com.
55.22.11.128.in-addr.arpa.      IN      PTR     newyork.cities.dec.com.
; %HOSTS_END%
```

B.10 Sample networks.db File

The following is a sample `networks.db` file:

```
$origin networks.cities.dec.com.
@  HS   SOA   chicago.cities.dec.com. postmaster.chicago.cities.dec.com. (
              13      ; Serial
              300     ; Refresh - 5 minutes
              60      ; Retry - 1 minute
              1209600 ; Expire - 2 weeks
              43200 ) ; Minimum - 12 hours
   HS   NS    chicago.cities.dec.com.
;
; %NETWORKS_START% - entries added by /var/dss/namedb/bin/make_networks
;
; Internet networks
;
loop            HS      TXT     "loop:127:loopback"
"127"           HS      CNAME   loop
"networks-0"    HS      CNAME   loop
ethernet        HS      TXT     "ethernet:128.11:mainnet"
"128.11"        HS      CNAME   ethernet
"networks-1"    HS      CNAME   ethernet
; %NETWORKS_END%
```

B.11 Sample passwd.db File

The following is a sample `passwd.db` file:

```
$origin passwd.cities.dec.com.
@   HS   SOA   chicago.cities.dec.com. postmaster.chicago.cities.dec.com. (
               13      ; Serial
               60      ; Refresh - 1 minute
               60      ; Retry - 1 minute
               1209600 ; Expire - 2 weeks
               3600 )  ; Minimum - 1 hour
    HS   NS    chicago.cities.dec.com.
;
; %PASSWD_START% - entries added by /var/dss/namedb/bin/make_passwd
nobody          HS      TXT     "nobody:*:-2:-2:Anonymous NFS User :/:"
"-2"            HS      CNAME   nobody
"passwd-0"      HS      CNAME   nobody
daemon          HS      TXT     "daemon:*:1:1:Mr Background:/:"
"1"             HS      CNAME   daemon
"passwd-1"      HS      CNAME   daemon
sys             HS      TXT     "sys:PASSWORD HERE:2:3:Mr Kernel:/usr/sys:"
"2"             HS      CNAME   sys
"passwd-2"      HS      CNAME   sys
; %PASSWD_END%
```

B.12 Sample protocols.db File

The following is a sample `protocols.db` file:

```
$origin protocols.cities.dec.com.
@   HS   SOA   chicago.cities.dec.com. postmaster.chicago.cities.dec.com. (
               13      ; Serial
               300     ; Refresh - 5 minutes
               60      ; Retry - 1 minute
               1209600 ; Expire - 2 weeks
               43200 ) ; Minimum - 12 hours
    HS   NS    chicago.cities.dec.com.
```

```
;
; %PROTOCOLS_START% - entries added by /var/dss/namedb/bin/make_protocols
;
; @(#)protocols 4.1 11/23/87
; Internet (IP) protocols
;
ip              HS      TXT     "ip:0:IP"
"0"             HS      CNAME   ip
"protocols-0"   HS      CNAME   ip
icmp            HS      TXT     "icmp:1:ICMP"
"1"             HS      CNAME   icmp
"protocols-1"   HS      CNAME   icmp
ggp             HS      TXT     "ggp:3:GGP"
"3"             HS      CNAME   ggp
"protocols-2"   HS      CNAME   ggp
; %PROTOCOLS_END%
```

B.13 Sample rpc.db File

The following is a sample `rpc.db` file:

```
$origin rpc.cities.dec.com.
@   HS   SOA   chicago.cities.dec.com. postmaster.chicago.cities.dec.com. (
               27      ; Serial
               300     ; Refresh - 5 minutes
               60      ; Retry - 1 minute
               1209600 ; Expire - 2 weeks
               43200 ) ; Minimum - 12 hours
    HS   NS    chicago.cities.dec.com.
;
; %RPC_START% - entries added by /var/dss/namedb/bin/make_rpc
selection_svc   HS      TXT     "selection_svc:100015:selnsvc"
"100015"        HS      CNAME   selection_svc
"rpc-0"         HS      CNAME   selection_svc
sprayd          HS      TXT     "sprayd:100012:spray"
"100012"        HS      CNAME   sprayd
"rpc-1"         HS      CNAME   sprayd
rquotad         HS      TXT     "rquotad:100011:rquotaprog:quota:rquota"
"100011"        HS      CNAME   rquotad
"rpc-2"         HS      CNAME   rquotad
; %RPC_END%
```

B.14 Sample services.db File

The following is a sample `services.db` file:

```
$origin services.cities.dec.com.
@   HS   SOA   chicago.cities.dec.com. postmaster.chicago.cities.dec.com. (
               13      ; Serial
               300     ; Refresh - 5 minutes
               60      ; Retry - 1 minute
               1209600 ; Expire - 2 weeks
               43200 ) ; Minimum - 12 hours
    HS   NS    chicago.cities.dec.com.
;
; %SERVICES_START% - entries added by /var/dss/namedb/bin/make_services
;
echo            HS      TXT     "echo:7:"
echo/udp        HS      TXT     "echo:7:udp:ping"
echo/tcp        HS      TXT     "echo:7:tcp:ping"
"7"             HS      CNAME   echo
"7/udp"         HS      CNAME   echo/udp
```

```
"7/tcp"          HS      CNAME   echo/tcp
"services-0"     HS      CNAME   echo/udp
"services-1"     HS      CNAME   echo/tcp
discard          HS      TXT     "discard:9:"
discard/udp      HS      TXT     "discard:9:udp:sink:null"
discard/tcp      HS      TXT     "discard:9:tcp:sink:null"
"9"              HS      CNAME   discard
"9/udp"          HS      CNAME   discard/udp
"9/tcp"          HS      CNAME   discard/tcp
"services-2"     HS      CNAME   discard/udp
"services-3"     HS      CNAME   discard/tcp
; %SERVICES_END%
```

B.15 The named_dump.db File

If you cause the `named` daemon to dump its cache by sending it a `SIGINT` signal, the results are stored in the `/var/tmp/named_dump.db` file.

This file is helpful in checking the BIND data files for possible errors.

B.16 The named.run File

If you turn on debugging for the `named` daemon, the results are recorded in the `/var/tmp/named.run` file. This file is helpful in troubleshooting the BIND service.

The nslookup Command C

This appendix provides sample interactive sessions with the `nslookup` command. These samples are intended to help you get started using the `nslookup` command. Here are the tasks shown in this appendix:

- Getting `nslookup` help
- Seeing which `nslookup` options are set
- Listing hosts in a domain
- Finding mail exchangers
- Finding the start of authority (SOA)
- Finding servers for a domain
- Obtaining a debug trace

C.1 Getting nslookup Help

To see a list of the `nslookup` commands, type a question mark (?) at the `nslookup` prompt:

```
# nslookup
Default Server:  localhost.cities.dec.com
Address:  127.0.0.1

> ?
Commands:           (identifiers are shown in uppercase, [] means optional)
NAME                - print info about the host/domain NAME using default server
NAME1 NAME2         - same as above, but use NAME2 as server
help or ?           - print help information
set OPTION          - set an option
    all             - print options, current server and host
    ALL             - print options, current server and host, state info
    [no]debug       - print debugging information
    [no]d2          - print exhaustive debugging information
    [no]defname     - append domain name to each query
    [no]recurse     - ask for recursive answer to query
    [no]vc          - always use a virtual circuit
    domain=NAME     - set default domain name to NAME
    root=NAME       - set root server to NAME
    retry=X         - set number of retries to X
    timeout=X       - set time-out interval to X
    class=X         - set class to X; IN, HS, or ANY
    [query]type=X - set query type to X; A, CNAME, MX, NS, PTR, SOA, TXT, or WKS
server NAME         - set default server to NAME, using current default server
lserver NAME        - set default server to NAME, using initial server
finger [NAME]       - finger the optional NAME
root                - set current default server to the root
ls [-adhlmst] DOMAIN [> FILE] - list DOMAIN, optional output to FILE
                    -a = list CNAME entries
                    -d = list all entries
```

```
                    -h = list HINFO entries
                    -l = list all entries
                    -m = list MX entries
                    -s = list WKS entries
                    -t = list TXT entries
view FILE           - sort an 'ls' output file and view it with more
```

C.2 Seeing Which nslookup Options Are Set

To see which `nslookup` options are set, use the `set all` command:

```
# nslookup
Default Server:  localhost.cities.dec.com
Address:  127.0.0.1

> set all
Default Server:  localhost.cities.dec.com
Address:  127.0.0.1

Set options:
  debug     defname    search     recurse    novc
  querytype=A     class=IN   timeout=4       retry=4
  domain=cities.dec.com
  search list: cities.dec.com dec.com
  root=nic.ddn.mil
```

C.3 Listing Hosts in a Domain

The following example shows how to use the `nslookup` command to create a file listing the hosts in the domain `cities.dec.com`, and to then view that file:

```
# nslookup
Default Server:  localhost.cities.dec.com
Address:  127.0.0.1

> ls cities.dec.com > filename
[wepel.cities.dec.com]
##########
Received 531 records.
> view filename
amherst                     128.67.45.1
ayers                       128.67.42.2
berlin                      128.67.45.3
boston                      128.67.45.4
cannes                      128.67.45.5
chandler                    128.67.45.6
chicago                     128.67.45.7
denver                      128.67.46.8
galway                      128.67.45.9
hollis                      128.67.49.10
ipswich                     128.67.45.11
laconia                     128.67.48.12
london                      128.67.45.13
madrid                      128.67.45.14
mason                       128.67.45.15
milford                     128.67.46.16
nashua                      128.67.45.17
newyork                     128.67.45.18
--More-- <RETURN>
paris                       128.67.42.19
phoenix                     128.67.46.20
tempe                       128.67.45.21
```

```
temple                          128.67.45.22
wilton                          128.67.45.23
<CTRL/c>
> <CTRL/d>
#
```

C.4 Finding Mail Exchangers

The following example shows how to use the `nslookup` command to find the mail exchanger for any system in the domain `cities.dec.com`. Note the use of a bogus host name. In the following example, the bogus host name is `nohost`:

```
# nslookup
Default Server:  localhost.cities.dec.com
Address:  127.0.0.1

> set type=mx
> nohost
Server:  localhost.cities.dec.com
Address:  127.0.0.1

nohost.cities.dec.com     pref = 51, mail exchanger = noun.cities.dec.com
nohost.cities.dec.com     pref = 50, mail exchanger = wepel.cities.dec.com
noun.cities.dec.com inet address = 128.45.45.79
wepel.cities.dec.com inet address = 128.45.45.93
> wepel
Server:  localhost.cities.dec.com
Address:  127.0.0.1

cities.dec.com origin = wepel.cities.dec.com
     mail addr = doe.wepel.cities.dec.com
     serial=10, refresh=1800, retry=3600, expire=1209600, min=86400
```

C.5 Finding the Start of Authority

The following sample session shows how to use the `nslookup` command to find the start of authority for the hosts named `wepel` and `decwrl.dec.com`:

```
# nslookup
Default Server:  localhost.cities.dec.com
Address:  127.0.0.1

> set type=SOA
> wepel
Server:  localhost.cities.dec.com
Address:  127.0.0.1

cities.dec.com origin = wepel.cities.dec.com
     mail addr = doe.wepel.cities.dec.com
     serial=10, refresh=1800, retry=3600, expire=1209600, min=86400
> decwrl.dec.com.
Server:  localhost.cities.dec.com
Address:  127.0.0.1

dec.com   origin = decwrl.dec.com
     mail addr = postmaster.decwrl.dec.com
     serial=197, refresh=43200, retry=3600, expire=1209600, min=86400
> <CTRL/d>
```

C.6 Looking at Hesiod Information in a Domain

If you have set up Hesiod on your ULTRIX system and would like to look at this information, you must use the `set class=`*value* and `set querytype=`*value* commands where value is HS and TXT respectively.

The following example presumes that the `networks` database is set up to be distributed with BIND/Hesiod. The answer received from the `nslookup` is that 128.45 is the network number for the network named `ethernet` in the networks.dec.com domain.

```
# nslookup
Default Server:   localhost.dec.com
Address: 127.0.0.1

> set cl=hs
> set q=txt
> ethernet.networks
Server:   localhost.dec.com
Address:   127.0.0.1

ethernet.networks.dec.com    ethernet:128.45
>
```

C.7 Finding Servers for a Domain

The following example shows how to use the `nslookup` command to find the servers for the domain `mit.edu.`:

```
# nslookup
Default Server:  localhost.cities.dec.com
Address:  127.0.0.1

> server nic.ddn.mil.
Default Server:  nic.ddn.mil
Address:  26.0.0.73

> set domain=mit.edu.
> ls
Server:  nic.ddn.mil
Address:  26.0.0.73

Name:    ls.mit.edu.
Served by:
- MIT-STRAWB.ARPA
        18.71.0.151
        MIT.EDU
- W20NS.MIT.EDU
        18.70.0.160
        MIT.EDU
- BITSY.MIT.EDU
        18.72.0.3
        MIT.EDU
- LITHIUM.LCS.MIT.EDU
        18.26.0.121
        MIT.EDU
> <CTRL/d>
#
```

C.8 Obtaining a Debug Trace

The following example shows how to use the `nslookup` command to help debug the BIND service:

```
# nslookup
Default Server:  localhost.cities.dec.com
Address:  127.0.0.1

> set debug
> set d2
> foobar
Server:  localhost.cities.dec.com
Address:  127.0.0.1

res_mkquery(0, foobar.cities.dec.com, 1, 1)
------------
SendRequest()
    HEADER:
     opcode = QUERY, id = 1, rcode = NOERROR
     header flags:  query, want recursion
     questions = 1,  answers = 0,  n.s. = 0,  additional = 0


    QUESTIONS:
     foobar.cities.dec.com, type = A, class = IN

------------
------------
Got answer:
    HEADER:
     opcode = QUERY, id = 1, rcode = NOERROR
     header flags:  resp, auth. answer, want recursion, recursion avail.
     questions = 1,  answers = 0,  n.s. = 0,  additional = 1

    QUESTIONS:
     foobar.cities.dec.com, type = A, class = IN
    ADDITIONAL RECORDS:
    ->  cities.dec.com
     type = SOA, class = IN, ttl = 86400, dlen = 37
     origin = wepel.cities.dec.com
     mail addr = doe.wepel.cities.dec.com
     serial=10, refresh=1800, retry=3600, expire=1209600, min=86400

------------
Name:    foobar.cities.dec.com

> noun
Server:  localhost.cities.dec.com
Address:  127.0.0.1

res_mkquery(0, noun.cities.dec.com, 1, 1)
------------
SendRequest()
    HEADER:
     opcode = QUERY, id = 2, rcode = NOERROR
     header flags:  query, want recursion
     questions = 1,  answers = 0,  n.s. = 0,  additional = 0

    QUESTIONS:
     noun.cities.dec.com, type = A, class = IN

------------
------------
Got answer:
```

```
    HEADER:
     opcode = QUERY, id = 2, rcode = NOERROR
     header flags:  resp, auth. answer, want recursion, recursion avail.
     questions = 1,  answers = 1,  n.s. = 0,  additional = 0

    QUESTIONS:
     noun.cities.dec.com, type = A, class = IN
    ANSWERS:

    ->  noun.cities.dec.com
     type = A, class = IN, ttl = 86400, dlen = 4
     inet address = 128.45.45.79

------------
Name:    noun.cities.dec.com
Address:  128.45.45.79

> set type=SOA
> noun
Server:  wepel.cities.dec.com
Address:  0.0.0.0

res_mkquery(0, noun.cities.dec.com, 1, 6)
------------
SendRequest()
    HEADER:
     opcode = QUERY, id = 3, rcode = NOERROR
     header flags:  query, want recursion
     questions = 1,  answers = 0,  n.s. = 0,  additional = 0

    QUESTIONS:
     noun.cities.dec.com, type = SOA, class = IN

------------
------------
Got answer:
    HEADER:
     opcode = QUERY, id = 3, rcode = NOERROR
     header flags:  resp, auth. answer, want recursion, recursion avail.
     questions = 1,  answers = 0,  n.s. = 0,  additional = 1

    QUESTIONS:
     noun.cities.dec.com, type = SOA, class = IN
    ADDITIONAL RECORDS:
    ->  cities.dec.com
     type = SOA, class = IN, ttl = 86400, dlen = 37
     origin = wepel.cities.dec.com
     mail addr = doe.wepel.cities.dec.com
     serial=10, refresh=1800, retry=3600, expire=1209600, min=86400

------------
cities.dec.com
     type = SOA, class = IN, ttl = 86400, dlen = 37
     origin = wepel.cities.dec.com
     mail addr = doe.wepel.cities.dec.com
     serial=10, refresh=1800, retry=3600, expire=1209600, min=86400

> decwrl.dec.com.
Server:  localhost.cities.dec.com
Address:  127.0.0.1

res_mkquery(0, decwrl.dec.com, 1, 6)
------------
SendRequest()
    HEADER:
```

```
     opcode = QUERY, id = 4, rcode = NOERROR
     header flags:  query, want recursion
     questions = 1,  answers = 0,  n.s. = 0,  additional = 0

    QUESTIONS:
     decwrl.dec.com, type = SOA, class = IN

------------
------------
Got answer:
    HEADER:
     opcode = QUERY, id = 4, rcode = NOERROR
     header flags:  resp, auth. answer, want recursion, recursion avail.
     questions = 1,  answers = 0,  n.s. = 0,  additional = 1

    QUESTIONS:
     decwrl.dec.com, type = SOA, class = IN
    ADDITIONAL RECORDS:
    ->  dec.com
     type = SOA, class = IN, ttl = 83633, dlen = 35
     origin = decwrl.dec.com
     mail addr = postmaster.decwrl.dec.com
     serial=197, refresh=43200, retry=3600, expire=1209600, min=86400

------------
dec.com
     type = SOA, class = IN, ttl = 83633, dlen = 35
     origin = decwrl.dec.com
     mail addr = postmaster.decwrl.dec.com
     serial=197, refresh=43200, retry=3600, expire=1209600, min=86400
> <CTRL/d>
#
```

BIND Questionnaire D

This appendix provides a copy of the BIND questionnaire that you need to complete and send to the NIC domain registrar to register your BIND domain. To obtain an on-line copy of the questionnaire, you can use the `ftp` command.

The following example shows a successful `ftp` exchange. In this example the site `nic.ddn.mil` is opened, the `help` option is invoked, and the BIND domain registration questionnaire is copied to the file `/tmp/questionnaire` on the local system:

```
# ftp
ftp> open
(to) nic.ddn.mil
Connected to nic.ddn.mil.
220 NIC.DDN.MIL FTP Server Process 5Z(47)-6 at Fri 10-Jun-88 12:07-PDT
Name (nic.ddn.mil:liza): anonymous
Password (nic.ddn.mil:anonymous):
331 ANONYMOUS user ok, send real ident as password.
230 User ANONYMOUS logged in at Fri 10-Jun-88 12:07-PDT

ftp> help
Commands may be abbreviated.  Commands are:

!               dir             mget            quit            trace
append          form            mkdir           quote           type
ascii           get             mls             recv            user
bell            glob            mode            binary          hash
cd              lcd             prompt          send            ?

ftp> get
(remote-file) netinfo:domain-template.txt
(local-file) /tmp/questionnaire
200 Port 4.30 at host 128.45.45.93 accepted.
150 ASCII retrieve of <NETINFO>DOMAIN-TEMPLATE.TXT.28 started.
226 Transfer completed. 6129 (8) bytes transferred.
6129 bytes received in 4.62 seconds (1.3 Kbytes/s)
ftp> close
221 QUIT command received. Goodbye.
ftp> bye
```

Upon completing a successful `ftp` exchange, as shown in the previous example, here is what you receive:

```
# more /tmp/questionnaire
```

[NETINFO:DOMAIN-TEMPLATE.TXT] [2/88]

To establish a domain, the following information must be sent to the NIC Domain Registrar (HOSTMASTER@NIC.DDN.MIL). Questions may be addressed to the NIC Hostmaster by electronic mail at the above address, or by phone at (415) 859-5539 or (800) 235-3155.

NOTE: The key people must have electronic mailboxes and NIC

"handles," unique NIC database identifiers. If you have access to WHOIS, please check to see if you are registered and if so, make sure the information is current. Include only your handle and any changes (if any) that need to be made in your entry. If you do not have access to "WHOIS", please provide all the information indicated and a NIC handle will be assigned.

(1) The name of the top-level domain to join.

For example: COM

(2) The NIC handle of the administrative head of the organization. Alternately, the person's name, title, mailing address, phone number, organization, and network mailbox. This is the contact point for administrative and policy questions about the domain. In the case of a research project, this should be the principal investigator.

For example:

Administrator

Organization The NetWorthy Corporation
Name Penelope Q. Sassafrass
Title President
Mail Address The NetWorthy Corporation
4676 Andrews Way, Suite 100
Santa Clara, CA 94302-1212
Phone Number (415) 123-4567
Net Mailbox Sassafrass@ECHO.TNC.COM
NIC Handle PQS

(3) The NIC handle of the technical contact for the domain. Alternately, the person's name, title, mailing address, phone number, organization, and network mailbox. This is the contact point for problems concerning the domain or zone, as well as for updating information about the domain or zone.

For example:

Technical and Zone Contact

Organization The NetWorthy Corporation
Name Ansel A. Aardvark
Title Executive Director
Mail Address The NetWorthy Corporation
4676 Andrews Way, Suite 100
Santa Clara, CA. 94302-1212
Phone Number (415) 123-6789
Net Mailbox Aardvark@ECHO.TNC.COM
NIC Handle AAA2

(4) The name of the domain (up to 12 characters). This is the name that will be used in tables and lists associating the domain with the domain server addresses. [While, from a technical standpoint, domain

names can be quite long (programmers beware), shorter names are easier for people to cope with.]

For example: TNC

(5) A description of the servers that provide the domain service for translating names to addresses for hosts in this domain, and the date they will be operational.

A good way to answer this question is to say "Our server is supplied by person or company X and does whatever their standard issue server does."

For example: Our server is a copy of the one operated by the NIC; it will be installed and made operational on 1 November 1987.

(6) Domains must provide at least two independent servers for the domain. Establishing the servers in physically separate locations and on different PSNs is strongly recommended. A description of the server machine and its backup, including

(a) Hardware and software (using keywords from the Assigned Numbers RFC).

(b) Host domain name and network addresses (which host on which network for each connected network).

(c) Any domain-style nicknames (please limit your domain-style nickname request to one)

For example:

- Hardware and software

  VAX-11/750 and UNIX, or
  IBM-PC and MS-DOS, or
  DEC-1090 and TOPS-20

- Host domain names and network addresses

  BAR.FOO.COM 10.9.0.193 on ARPANET

- Domain-style nickname

  BR.FOO.COM (same as BAR.FOO.COM 10.9.0.13 on ARPANET)

(7) Planned mapping of names of any other network hosts, other than the server machines, into the new domain's naming space.

For example:

BAR-FOO2.ARPA (10.8.0.193) -> FOO2.BAR.COM
BAR-FOO3.ARPA (10.7.0.193) -> FOO3.BAR.COM

BAR-FOO4.ARPA (10.6.0.193) -> FOO4.BAR.COM

(8) An estimate of the number of hosts that will be in the domain.

(a) Initially
(b) Within one year
(c) Two years
(d) Five years.

For example:

(a) Initially = 50
(b) One year = 100
(c) Two years = 200
(d) Five years = 500

(9) The date you expect the fully qualified domain name to become the official host name in HOSTS.TXT.

Please note: Registration of this domain does not imply an automatic name change to previously registered ARPANET or MILNET hosts that will be included in this domain. If changing to a fully qualified domain name (e.g., FOO.BAR.COM) causes a change in the official host name of an ARPANET or MILNET host, DCA approval must be obtained. This should be done after your domain name is approved by Hostmaster. Allow 10 working days for your requested changes to be processed. ARPANET (network 10) sites should contact ARPANETMGR@DDN1.ARPA. MILNET (network 26) sites should contact MILNETMGR@DDN1.ARPA.

(10) Please describe your organization briefly.

For example: The NetWorthy Corporation is a consulting organization of people working with UNIX and the C language in an electronic networking environment. It sponsors two technical conferences annually and distributes a bimonthly newsletter.

References E

This appendix lists the papers, articles, and RFCs associated with the BIND service that you may find useful. You can obtain the RFCs online by using the `ftp` command as shown in Appendix C. See `ftp`(1c) in the *ULTRIX Reference Pages* for further information.

[Dunlap 86a] Dunlap, K. J., Bloom, J. M., "Experiences Implementing BIND, A Distributed Name Server for the DARPA Internet," Proceedings USENIX Summer Conference, Atlanta, Georga. June 1986, pages 172-181

[Dunlap 86b] Dunlap, K. J., "Name Server Operations Guide for BIND," Unix System Manager's Manual, SMM-11. 4.3 Berkeley Software Distribution, Virtual VAX-11 Version. University of California. April 1986

[Dyer 87] Dyer, S., and F. Hsu, "Hesiod," Project Athena Technical Plan - Name Service, April 1987, version 1.9.

[IEN-116] Postel J., "Internet Name Server," IEN-116, USC/Information Sciences Institute, August 1979.

[Mockapetris 88] Mockapetris, P. V., Dunlap, K. J., "Development of the Domain Name System," Proceedings ACM SIGCOMM 1988 Symposium, Stanford University, Stanford, California, August 1988.

[Quarterman 86] Quarterman, J., and J. Hoskins, "Notable Computer Networks," Communications of the ACM, October 1986, volume 29, number 10.

[RFC-882] P. Mockapetris, "Domain names - Concepts and Facilities," RFC-882, USC/Information Sciences Institute, November 1983.

[RFC-883] P. Mockapetris, "Domain names - Implementation and Specification," RFC-883, USC/Information Sciences Institute, November 1983.

[RFC-920] J. Postel and J. Reynolds, "Domain Requirements," RFC-920, USC/Information Sciences Institute October 1984.

[RFC-973] P. Mockapetris, "Domain System Changes and Observations," RFC-973, USC/Information Sciences Institute, January 1986.

[RFC-974] C. Partridge, "Mail routing and the domain system," RFC-974, CSNET CIC BBN Labs, January 1986.

[RFC-1031] W. Lazear, "MILNET Name Domain Transition," RFC-1031, November 1987.

[RFC-1032] M. K. Stahl, "Establishing a Domain - Guidelines for Administrators," RFC-1032, November 1987.

[RFC-1033] M. K. Lottor, "Domain Administrators Operations Guide," RFC-1033, November 1987.

[RFC-1034] Mockapetris, P. V., "Domain Names - Concepts and Facilities," RFC 1034, USC/Information Sciences Institute, November 1987.

[RFC-1035] Mockapetris, P. V., "Domain names - Implementation and Specification," RFC 1035, USC/Information Sciences Institute, November 1987.

Note

In the references listed, *RFC* refers to papers in the ARPA Request for Comments series and *IEN* refers to ARPA Internet Experiment Notes. Both the RFCs and IENs may be obtained from the Network Information Center, SRI International, Menlo Park, CA 94025, or from the authors of the papers.

Index

A

address data file entry
defined, 2–3

B

Berkeley Internet Name Domain
See BIND service
BIND query
resolving, 1–7
BIND server
root, 1–4
BIND service
See also BIND/Hesiod service
defined, 1–1
further information, 4–7
managing, 4–7
relationship to Hesiod, 1–1
resolver, 1–1
server, 1–1
two parts, 1–1
BIND/Hesiod
maintaining domains, 4–1
BIND/Hesiod client
automatic setup, 3–4
defined, 1–7
manual setup, A–1 to A–2
BIND/Hesiod data file entries, 2–1 to 2–10
A, 2–3
CNAME, 2–4
HINFO, 2–4
include, 2–3
MB, 2–5
MG, 2–5
BIND/Hesiod data file entries (cont.)
MINFO, 2–6
MR, 2–6
MX, 2–7
NS, 2–7
origin, 2–3
PTR, 2–8
SOA, 2–8
TXT, 2–9
WKS, 2–10
BIND/Hesiod file entry
defined, 2–1
format of, 2–1 to 2–10
BIND/Hesiod manual setup
creating the hesiod.conf file, A–2
creating the resolv.conf file, A–1
editing the hosts file, A–2
BIND/Hesiod primary server
creating database files, 3–6n
BIND/Hesiod query
resolving, 1–7 to 1–8
BIND/Hesiod server
See also caching server
See also forwarding server
See also master server
See also root server
See also slave server
automatic setup
primary, 3–6 to 3–8
secondary, 3–8 to 3–9
slave, 3–8 to 3–9
caching, 1–5
defined, 1–3
forwarding, 1–5 to 1–6

BIND/Hesiod server (cont.)
manual setup, A–2 to A–5
master, 1–4 to 1–5
root, 1–4
slave, 1–6
BIND/Hesiod service
and the svc.conf file, 3–1
configuring a BIND caching server, 3–9
failure causes, 5–1
introduction, 1–1 to 1–8
making a Hesiod database, 4–7
managing, 4–1
removing with bindsetup, 3–11
setting up manually, A–1 to A–5
setting up with bindsetup, 2–10
troubleshooting, 5–1 to 5–6
with no forwarder, 1–6
bindsetup command
command line, 3–5e
running, 3–4, 3–6, 3–8
BITNET network
contacting, 4–3
boot file
editing, A–2 to A–3
sample caching server, B–5
sample primary server, B–3
sample secondary server, B–4
sample slave server, B–4

C

cache file
sample, B–5
caching server
defined, 1–5
canonical name
defined, 2–4
CNAME data file entry
defined, 2–4
CSNET network
contacting, 4–3

D

DARPA network
contacting, 4–3
data file
updating, 4–4
data file directory
default, 3–6
data file entry
address, 2–3
CNAME, 2–4
HINFO, 2–4
include, 2–3
MB, 2–5
MG, 2–5
MINFO, 2–6
MR, 2–6
MX, 2–7
NS, 2–7
origin, 2–3
SOA, 2–8
WKS, 2–10
debug files
reviewing, 5–2
domain
case insensitive, 4–2n
defined, 4–1, 4–2
fully qualified name, 4–2
maintaining, 4–1 to 4–2
naming, 4–2
relative name, 4–2
subdomain of, 4–2
domain administrator
defined, 4–1
duties of, 4–1
domain hierarchy, 1–1, 1–2f, 1–3f
label, 1–1
leaf domain, 1–2
root, 1–1
top-level domain, 1–2
domain name
trailing period, 3–5n

F

forwarding server
defined, 1–5
ftp command, D–1e
fully qualified name
defined, 1–3

G

gethostbyname routine
with BIND, 1–7

H

Hesiod
making a Hesiod database, 4–7
Hesiod name service
relationship to BIND, 1–1
Hesiod query
resolving, 1–8
Hesiod Text Entry
defined, 2–9
format of, 2–9 to 2–10
hesiod.conf file
creating, A–2
HINFO data file entry
defined, 2–4
host
naming, 4–2
host name
obtaining, 4–5
hosts file
editing, A–2
hosts.db file
defined, A–3

I

include data file entry
defined, 2–3
IP address
obtaining, 4–5

K

Kerberos, 3–8
and BIND/Hesiod caching server, 3–10
and BIND/Hesiod primary server, 3–7

M

master server
defined, 1–4
MB data file entry
defined, 2–5
MG data file entry
defined, 2–5
MINFO data file entry
defined, 2–6
MR data file entry
defined, 2–6
MX data file entry
bogus name, 4–5
defined, 2–7

N

named daemon
inetd, 5–2n
obtaining PID, 5–5
rc.local entry, A–4
for Kerberos, A–4
sending signals to, 5–5 to 5–6
named.boot file
defined, B–3
named.ca file
defined, A–3
named_dump file
reviewing, 5–3
named.local file
defined, A–3
named.rev file
defined, A–3
named.run file
reviewing, 5–4
named.stats file
reviewing, 5–4

NIC
address, 1–4
phone number, 1–4
NIC whois service
See whois service
NS data file entry
defined, 2–7
nslookup command
debug trace, C–5
finding MX, C–3, C–3e
finding servers, C–4, C–4e
finding SOA, C–3, C–3e
getting debug trace, C–5e to C–7e
getting help, C–1, C–1e
host info, 4–5
listing hosts, C–2, C–2e to C–3e
obtaining IP info, 4–5
viewing options, C–2
nsquery command
obtaining host info, 4–5
obtaining IP info, 4–5

O

origin data file entry
defined, 2–3

P

PTR data file entry
defined, 2–8
public networks
registering with, 4–3 to 4–4

Q

questionnaire (bind)
sample of, D–1 to D–4

R

removing the BIND/Hesiod service, 3–11
resolv.conf file
entries of, A–1
reviewing, 5–2
resolver
See also resolv.conf file
resolver file
creating, A–1
resource record
See data file
See data file entries
defined, 2–1
root server
defined, 1–4
setting up, B–1n
root servers
list of, 1–4

S

service order file
See also svc.conf files
services file
specifying port, 1–1
services order file
See svc.conf file
setting up with svcsetup, 3–2
slave server
defined, 1–6
SOA data file entry
defined, 2–8
svc.conf file
manually editing, 3–4
recommended service order, 3–4
service order restrictions, 3–4
svcsetup command
defined, 3–2
syslog file
reviewing, 5–3
sample of, 5–3

T

technical and zone contact
defined, 4–2
top-level domain
country, 4–2
registering, 4–2

trailing period
significance of, 1–1

W

whois service
using, 4–6
WKS data file entry
defined, 2–10

Z

zone
defined, 1–2, 4–2

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-234-1998 using a 1200- or 2400-baud modem from anywhere in the USA, Canada, or Puerto Rico. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital Subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	————	Local Digital subsidiary or approved distributor
Internal*	————	SSB Order Processing - WMO/E15 *or* Software Supply Business Digital Equipment Corporation Westminster, Massachusetts 01473

* For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

ULTRIX
Guide to the BIND/Hesiod Service
AA-LY21B-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

What would you like to see more/less of? _____

_____

_____

What do you like best about this manual? _____

_____

_____

What do you like least about this manual? _____

_____

_____

Please list errors you have found in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

What version of the software described by this manual are you using? _____

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Email _____ Phone _____

Do Not Tear – Fold Here and Tape

digital™

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST–CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZKO3–2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062–9987

Do Not Tear – Fold Here

Cut
Along
Dotted
Line

Reader's Comments

ULTRIX
Guide to the BIND/Hesiod Service
AA-LY21B-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

What would you like to see more/less of? _____

_____

_____

What do you like best about this manual? _____

_____

_____

What do you like least about this manual? _____

_____

_____

Please list errors you have found in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

What version of the software described by this manual are you using? _____

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Email _____ Phone _____

Do Not Tear – Fold Here and Tape

digital™

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST–CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZKO3–2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062–9987

Do Not Tear – Fold Here

Cut
Along
Dotted
Line

digital